Command Reference

Adabas Version 7.4.2

Table of Contents

Command Reference 1
Command Reference 1
Adabas Command Overview and Syntax 2
Adabas Command Overview and Syntax 2
Adabas Command Overview 3
Adabas Command Overview 3
Command Types 3
Database Query Commands (Sx) 3
Data Storage Read Commands (L1-L6) 5
Associator Read Commands (L9, LF) 6
Database Modification Commands (A1, E1, N1/N2) 7
Logical Transaction Control Commands (ET/BT) 7
Special Commands 8
Transactions and ET Logic 10
What is a Logical Transaction? 11
Transaction Sequence Number 11
ET Transaction Time Limit 11
Back Out Transaction (BT) Command 12
Autobackout 12
End Transaction (ET) Command 13
User (ET) Data 13
Adabas User ID 14
Competitive Updating 14
ET User Record Hold 15
Avoiding Resource Deadlock 16
Exclusive Control Updating 17
Non-Activity Time Limit 18
For an ET Logic User 19
For an Exclusive File Control User (EXF) 19
For an Access Only User 19
Adabas Calling Procedure 20
Adabas Calling Procedure 20
Specifying an Adabas Call 20
Syntax Conventions 21
Variables and Constants 21
Optional Elements, Optional Values, and Default Values 22
Repeating Elements 22
Control Block 23
Control Block Fields 24
Programming Examples 29
Logging the Control Block 29
Format and Record Buffers 30
Format Buffer Syntax 30
format-selection-criterion 31
record-format 32
Format Buffer Performance Considerations 44
Record Buffer 45
Specifying and Reading the SQL Null Indicator 46

Specifying a Field with LA (Long Alpha) Option . . . 47
Format and Record Buffer Examples . . . 48
Example 1: Using Elementary Fields (Standard Length and Format) . . . 48
Example 2: Using Elementary Fields (Length and Format Override) . . . 49
Example 3: A Reference to a Periodic Group . . . 49
Example 4: The First Two Occurrences of Periodic Group GB . . . 49
Example 5: The Sixth Value of the Multiple-Value Field MF . . . 49
Example 6: The First Two Values of the Multiple-Value Field MF . . . 50
Example 7: The Highest Occurrence Number of a Periodic Group GC and the Existing Number of Values for the Multiple-Value Field MF . . . 50
Search and Value Buffers . . . 50
Search Buffer Syntax . . . 51
Overview . . . 51
Search Expression . . . 52
Connecting Search Expressions . . . 55
Physically Coupled Files . . . 57
Soft Coupling . . . 58
Value Buffer . . . 58
SQL Null Values and Indicators . . . 59
Sign Handling . . . 59
Search/Value Buffer Examples . . . 60
Example 1: Using a Single Search Expression . . . 60
Example 2: Using Search Expressions Connected by AND . . . 60
Example 3: Using Search Expressions Connected by OR . . . 61
Example 4: Using Search Expressions Connected by FROM-TO . . . 61
Example 5: Using Search Expression with BUT-NOT . . . 61
Example 6: Using a Multiple-Value Descriptor . . . 62
Example 7: Using a Descriptor Within a Periodic Group . . . 62
Example 8: Using a Subdescriptor . . . 62
Example 9: Using a Superdescriptor with Alphanumeric Format . . . 63
Example 10: Using a Superdescriptor with Binary Format . . . 63
Example 11: Using Previously Created ISN Lists . . . 63
Example 12: Using a Value Operator . . . 64
Example 13: Using Both Value and Connecting Operators . . . 64
Example 14: Using Physically Coupled Files . . . 64
Example 15: Using Single Soft Coupling Criterion and Single Search Criterion . . . 65
Example 16: Using Single Soft Coupling Criterion and Multiple Search Criteria . . . 65
Example 17: Using Multiple Soft Coupling Criteria and Multiple Search Criteria . . . 66
Example 18: Using Multiple Soft Coupling Criteria and Multiple Search Criteria with Physical Coupling . . . 66
ISN Buffer . . . 67
General Programming Considerations . . . 68
General Programming Considerations . . . 68
Command ID, Format ID, Global Format ID . . . 68
Command IDs for Read Sequential Commands . . . 69
Command/Format IDs for Read, Update, and Find Commands . . . 69
Using Separate Command ID and Format IDs . . . 70
Using a Global Format ID . . . 70
Control Block Settings for Command / Format / Global Format IDs . . . 71
Command IDs Used with ISN Lists . . . 71
Automatic Command ID Generation . . . 72

Releasing Command IDs 72
Internal Identification of Command IDs 72
Examples of Command ID Use 72
ISN List Processing 73
Storage of ISN Lists 73
Retrieval of ISN Lists 74
ISN List Processing Examples 75
Using the Multifetch/Prefetch Feature 77
Multifetching vs. Prefetching 78
Invoking Multifetch/Prefetch 78
Multifetch Operation Processing 79
Prefetch Operation Processing 81
Additional Prefetch Programming Considerations 82
Adabas Commands 83
Adabas Commands 83
A1 Command: Update Record 85
A1 Command: Update Record 85
Function and Use 85
Command: A1 85
Control Block 87
Buffers 89
Format Buffer 89
Record Buffer 89
Additional Considerations 89
Examples 94
Example 1 94
Example 2 94
BT Command: Back Out Transaction 96
BT Command: Back Out Transaction 96
Function and Use 96
Command: BT 96
Control Block 97
ISN Buffer 99
Examples 99
Example 1 99
Example 2 100
Example 3 100
C1 Command: Write a Checkpoint 101
C1 Command: Write a Checkpoint 101
Function and Use 101
Command: C1 101
Control Block 102
Example 103
C3 Command: Write Checkpoint 104
C3 Command: Write Checkpoint 104
Function and Use 104
Command: C3 104
Control Block 105
Record Buffer 106
Example 106

C5 Command: Write User Data to Protection Log . . . 107
C5 Command: Write User Data to Protection Log . . . 107
Function and Use . . . 107
Command: C5 . . . 107
Control Block . . . 108
Record Buffer . . . 108
Example . . . 109
CL Command: Close User Session . . . 110
CL Command: Close User Session . . . 110
Function and Use . . . 110
Command: CL . . . 110
Control Block . . . 112
Record Buffer . . . 113
Examples . . . 113
Example 1 . . . 113
Example 2 . . . 113
E1 Command: Delete Record / Refresh File . . . 114
E1 Command: Delete Record / Refresh File . . . 114
Function and Use . . . 114
Command: E1 . . . 114
Control Block . . . 115
Examples . . . 117
Example 1 . . . 117
Example 2 . . . 117
ET Command: End Transaction . . . 118
ET Command: End Transaction . . . 118
Function and Use . . . 118
Command: ET . . . 119
Control Block . . . 120
Buffers . . . 121
Record Buffer . . . 121
ISN Buffer . . . 122
Examples . . . 122
Example 1: ET without User Data . . . 122
Example 2: ET with User Data . . . 122
Example 3: ET with Hold ISN Option . . . 122
HI Command: Hold Record . . . 124
HI Command: Hold Record . . . 124
Function and Use . . . 124
Command: HI . . . 124
Control Block . . . 125
Example . . . 126
L1 / L4 Command: Read / Read and Hold Record . . . 127
L1 / L4 Command: Read / Read and Hold Record . . . 127
Function and Use . . . 127
Command: L1 / L4 . . . 128
Control Block . . . 129
Buffers . . . 133
Format Buffer . . . 133
Record Buffer . . . 133
ISN Buffer . . . 133

Examples . . . 133
Example 1: Reading a Single Record . . . 134
Example 2: Reading a Set of Records . . . 134
Example 3: Reading a Set of Records Using the GET NEXT Option . . . 135
Example 4: Read with Hold . . . 136
Example 5: Read Using the Read ISN Sequence Option . . . 137
Example 6: Reading Multiple-Value Fields and Periodic Groups . . . 137
L2 / L5 Command: Read Physical Sequential . . . 140
L2 / L5 Command: Read Physical Sequential . . . 140
Function and Use . . . 140
Command: L2 / L5 . . . 140
Control Block . . . 142
Buffers . . . 145
Format Buffer . . . 145
Record Buffer . . . 145
ISN Buffer . . . 145
Additional Considerations . . . 145
Examples . . . 145
L3 / L6 Command: Read Logical Sequential . . . 148
L3 / L6 Command: Read Logical Sequential . . . 148
Function and Use . . . 148
Repositioning to a Particular Value . . . 149
Changing the Direction of the Read . . . 149
Command: L3 / L6 . . . 149
Control Block . . . 151
Buffers . . . 157
Format Buffer . . . 157
Record Buffer . . . 157
Search Buffer . . . 157
Value Buffer . . . 158
Search and Value Buffer Examples . . . 159
ISN Buffer . . . 161
Additional Considerations . . . 161
Examples . . . 162
Example 1 . . . 162
Example 2: Read Logical Sequential with Record Hold . . . 163
Example 3: ASCENDING Option . . . 163
Example 4: DESCENDING Option with Repositioning . . . 164
L9 Command: Read Descriptor Values . . . 166
L9 Command: Read Descriptor Values . . . 166
Function and Use . . . 166
Changing the Direction of the Read . . . 166
Command: L9 . . . 167
Control Block . . . 168
Buffers . . . 172
Format Buffer . . . 172
Record Buffer . . . 173
Search Buffer . . . 173
Value Buffer . . . 175
ISN Buffer . . . 175
Additional Considerations . . . 176

Examples . . . 176
Example 1 . . . 176
Example 2 . . . 177
Example 3: Overview of Sequence Options . . . 177
LF Command: Read Field Definitions . . . 179
LF Command: Read Field Definitions . . . 179
Function and Use . . . 179
Command: LF . . . 179
Control Block . . . 180
Record Buffer . . . 182
Command Option 2 is Set to "S" . . . 182
Command Option 2 is Set to "I" . . . 188
Command Option 2 is Set to Neither S Nor I . . . 190
Example . . . 191
N1 / N2 Command: Add Record . . . 193
N1 / N2 Command: Add Record . . . 193
Function and Use . . . 193
Command: N1 / N2 . . . 193
Control Block . . . 195
Buffers . . . 197
Format Buffer . . . 197
Record Buffer . . . 200
Examples . . . 200
Example 1 . . . 200
Example 2 . . . 201
OP Command: Open User Session . . . 203
OP Command: Open User Session . . . 203
Function and Use . . . 203
User Types . . . 204
Access-Only Users . . . 204
Exclusive Control Users . . . 204
ET Logic Users . . . 204
Command: OP . . . 204
Control Block . . . 206
Record Buffer . . . 209
User Queue Element . . . 211
User Type and File Lists . . . 211
Special Encoding Information . . . 212
Exceeding Time Limits . . . 212
Values Returned in Control Block Fields . . . 212
Examples . . . 214
Example 1: Access-Only User . . . 214
Example 2: ET Logic User . . . 214
Example 3: Exclusive Control User Without ET Logic . . . 215
Example 4: Exclusive Control User with ET Logic . . . 215
Example 5: Special Encoding for Wide-Character Fields . . . 216
RC Command: Release Command ID or Global Format ID . . . 217
RC Command: Release Command ID or Global Format ID . . . 217
Function and Use . . . 217
Command: RC . . . 218
Control Block . . . 218

Examples . . . 220
Example 1 . . . 220
Example 2 . . . 221
Example 3 . . . 221
Example 4 . . . 221
Example 5 . . . 222
RE Command: Read ETuser Data . . . 223
RE Command: Read ETuser Data . . . 223
Function and Use . . . 223
Command: RE . . . 223
Control Block . . . 225
Record Buffer . . . 226
Examples . . . 226
Example 1 . . . 226
Example 2 . . . 227
Example 3 . . . 227
RI Command: Release Record . . . 228
RI Command: Release Record . . . 228
Function and Use . . . 228
Command: RI . . . 228
Control Block . . . 229
Examples . . . 230
Example 1 . . . 230
Example 2 . . . 230
S1 / S2 / S4 Command: Find Records . . . 231
S1 / S2 / S4 Command: Find Records . . . 231
Function and Use . . . 231
Command: S1 / S2 / S4 . . . 232
Control Block . . . 234
Buffers . . . 238
Format Buffer . . . 238
Record Buffer . . . 238
Search and Value Buffers . . . 238
ISN Buffer . . . 239
Examples . . . 239
S5 Command: Find Coupled ISNs . . . 247
S5 Command: Find Coupled ISNs . . . 247
Function and Use . . . 247
Command: S5 . . . 247
Control Block . . . 249
ISN Buffer . . . 251
Example . . . 251
S8 Command: Process ISN Lists . . . 252
S8 Command: Process ISN Lists . . . 252
Function and Use . . . 252
Command: S8 . . . 253
Control Block . . . 254
ISN Buffer . . . 256
Example . . . 256

S9 Command: Sort ISN Lists . . . 258
S9 Command: Sort ISN Lists . . . 258
Function and Use . . . 258
Command: S9 . . . 258
Control Block . . . 260
ISN Buffer . . . 263
Examples . . . 263
Programming Examples . . . 265
Programming Examples . . . 265
File Definitions Used in Examples . . . 266
File Definitions Used in Examples . . . 266
Examples for Assembler . . . 268
Examples for Assembler . . . 268
Example 1 . . . 268
Example 2 . . . 269
Example 3 : User Session with ET Logic . . . 270
Session Initialization . . . 271
Transaction Processing . . . 272
Examples for COBOL . . . 275
Examples for COBOL . . . 275
Example 1 . . . 276
Example 2 . . . 277
Example 3 : User Session with ET Logic . . . 278
Example for PL/I . . . 281
Example for PL/I . . . 281
Example 1 . . . 282
Example 2 . . . 283
Example 3 . . . 284
Example for FORTRAN . . . 287
Example for FORTRAN . . . 287
Example 1 . . . 287

Command Reference

This documentation describes the powerful and flexible set of Adabas direct call commands for performing database operations. These call commands provide a direct interface to the Adabas database when Natural or another fourth-generation database language is not being used.

Note:
Dataset names starting with DD are referred to in Adabas documentation with a slash separating the DD from the remainder of the dataset name to accommodate VSE/ESA dataset names that do not contain the DD prefix. The slash is not part of the dataset name.

The Adabas Command Reference documentation is organized in the following parts:

●	Adabas Command Overview and Syntax	Overview of Adabas commands and detailed description of the calling procedure with general programming considerations
●	Adabas Commands	Detailed description of Adabas commands
●	Programming Examples	Programming examples of direct Adabas calls in each host language

Adabas Command Overview and Syntax

This part of the Adabas Command Reference documentation provides an overview of the Adabas commands. The procedures used to call Adabas to execute an Adabas command are described and several concepts that are important to consider when programming calls are explained.

The information is organized under the following headings:

- Adabas Command Overview
- Adabas Calling Procedure
- General Programming Considerations

Adabas Command Overview

This section provides an overview of the Adabas commands categorized by function: database query, database modification, Data Storage read, Associator read, logical transaction processing, and "special" commands.

In addition, Adabas command facilities related to data protection, recovery, and user restart are described. The "transaction" concept is introduced and ET logic operations are explained. Competitive updating is discussed for ET logic (record hold/release and avoidance of resource deadlock) and exclusive control users; nonactivity timeouts are described for all user types.

This chapter covers the following topics:

- Command Types
- Transactions and ET Logic
- Competitive Updating
- Non-Activity Time Limit

Command Types

The commands can be categorized into the following functions:

- Database Query Commands (Sx)
- Data Storage Read Commands (L1-L6)
- Associator Read Commands (L9, LF)
- Database Modification Commands (A1, E1, N1/N2)
- Logical Transaction Control Commands (ET/BT)
- Special Commands

Database Query Commands (Sx)

Database query commands (S1/S4, S2, S5) search for and return the ISNs of specified records or record groups according to specified search criteria. Other commands in this category (S8, S9) sort the resulting ISN lists in preparation for later operations.

The ISN lists resulting from any Sx command may be saved on the Adabas Work for later retrieval during the user session.

In most cases, these commands do not actually read the database; ISNs are read directly from the Associator’s inverted lists. Options allow the ISN’s record to be placed in hold status to prevent its being updated by other programs until the record is released; if desired, additional field values contained in the first ISN’s record can be read from Data Storage.

This section covers the following database query commands:

- S1/S4
- S2
- S5
- S8
- S9

S1/S4

The S1/S4 command selects the records that satisfy given search criteria.

- If only descriptors are used, the query is resolved using the inverted list alone without accessing Data Storage.

- If no descriptors are included in the search criteria, the S1/4 command reads each record in Data Storage to resolve the query.

- If both descriptors and nondescriptors are used within the search criteria, Adabas first searches the inverted list for the descriptors and then reads the Data Storage for all matching ISNs to check the nondescriptors.

S1 and S4 commands return the count of records that satisfy the search criterion, and a list of the ISNs for those records. An option permits the record identified by the first ISN in the resulting ISN list to be read from Data Storage.

The S4 command may be used to place the record identified by the first ISN in the ISN list in hold status. This prevents another user from updating the record until it is released.

S2

The S2 command is equivalent to the S1 command except that the ISNs of the records selected are returned in the sort sequence of a user-specified descriptor (or descriptors). One to three descriptors may be used. Ascending or descending sequence may be specified.

S5

The S5 command is used to select the ISNs of records in one file which are coupled to a record with a given ISN in another file.

The user specifies the file and ISN for which coupled ISNs are to be returned and the file in which the coupled records are to be selected. Adabas returns the number of records coupled to the ISN and the list of the coupled ISNs.

S8

The S8 command performs logical (AND, OR, or NOT) operations on two ISN lists previously created by an S1/S4, S5, S8, or S9 command.

- AND results in a single ISN list containing ISNs present in both source lists.

- OR results in a single ISN list containing ISNs present in either source list.
- NOT results in a single ISN list containing ISNs present in the first list but not in the second list.

S9

The S9 command sorts an ISN list created previously by a S1/S4, S2, S5, S8 or S9 command.

The ISN list may be sorted by ISN (ascending sequence only), or by one to three user-specified descriptors (ascending or descending sequence).

Data Storage Read Commands (L1-L6)

The L1 through L6 commands are used to read actual records from Data Storage. Depending on the specified command and its options, records are read individually

- in the sequence in which they are stored;
- in the order of an ISN list created by one of the database query commands; or
- in logical sequence according to a user-specified descriptor.

A hold option allows the database records to be locked until released by a separate command or at transaction end.

This section covers the following data storage read commands:

- L1/L4
- L2/L5
- L3/L6

L1/L4

The L1 command reads a single record from Data Storage. The user specifies the file number, ISN of the record to be read, and the fields for which values are to be returned. Adabas returns the requested field values.

The L4 command is the same as the L1 command except that the record is placed in hold status. This prevents other users from updating the record until it is released.

The "multifetch/prefetch" option prefetches records on either a session or command basis. This can reduce the overhead for multiple record fetches. The multifetch option is platform-independent.

The "GET NEXT" option may be used to read one or more records identified by ISNs contained in an ISN list without the user having to specify each ISN. Usually, the ISN list is created by a previous Sx command.

The "response code" option issues a response code 145 if the L4 command cannot place a record in hold status because it is being held by another user. Otherwise, the user is placed in wait status until the ISN (and record) are freed or the waiting user’s transaction is timed out.

The "ISN sequence" option may be used to read records in ISN sequence. The ISN specified by the user is read, unless it is not present, in which case the record which has the next higher ISN is read.

L2/L5

The L2 command reads the records from a file in the sequence in which they are physically stored in Data Storage. The user specifies the file to be read and the fields for which values are to be returned. Adabas returns the requested field values.

The L5 command is the same as the L2 command except that the record read is placed in hold status. This prevents other users from updating the record until it is released. The multifetch/prefetch and response code options (see the L1/L4 command description, above) also apply to the L2/L5 commands.

L3/L6

The L3 command reads records from Data Storage in the logical sequence of a user-specified descriptor. The user specifies the file to be read, the descriptor to be used for sequence control, and the fields for which values are to be returned. Adabas returns the requested field values.

Command option 2 for the L3/L6 command specifies whether the records are read in ascending or descending order. In addition, command option 2 can be used to specify a start value.

The L6 command is the same as the L3 command except that the record read is placed in hold status. This prevents other users from updating the record until it is released.

In addition to the multifetch/prefetch and response code options (see the L1/L4 command description, above), the start value option permits reading to begin at a user-specified value and/or ISN.

Associator Read Commands (L9, LF)

The L9 and LF commands read information directly from the Associator inverted lists or field definition tables (FDTs), returning either the inverted list values for a specified descriptor or the field definitions for a specified file in the database.

This section covers the following associator read commands:

- L9
- LF

L9

The L9 command returns each value contained in the inverted list for a given descriptor, and the number of records in which the value is contained.

The user specifies the file and descriptor for which values are to be returned, the value at which the command is to begin, and whether the values are returned in ascending or descending sequence.

LF

The LF command returns the field definitions for a file. The user specifies the file for which the field definitions are to be returned.

The field definitions for all the fields in the file are returned. Each field definition consists of the field name, level number, standard format, standard length, and definition options.

Database Modification Commands (A1, E1, N1/N2)

Database modification commands (A1, E1, and N1/N2) add, change, or delete database records and update the related Associator lists accordingly. ISNs can be assigned to new records either by the user or by Adabas.

This section covers the following database modification commands:

- A1
- E1
- N1/N2

A1

The A1 command updates the contents of one or more fields within a record. The user specifies the file and ISN of the record to be updated, together with the fields to be updated and the values to be used for updating.

Adabas performs all necessary modifications to the Associator and Data Storage. Associator updating is required only if one or more descriptors are updated.

A hold option is available for the purpose of placing the record to be updated in hold status prior to the update.

E1

The E1 command deletes a record or refreshes a file. The user specifies the file and ISN of the record to be deleted, or specifies the file only (without an ISN and command ID) to refresh the file.

Adabas performs all necessary modifications to the Associator and Data Storage.

A hold option is available for the purpose of placing the record to be deleted in hold status prior to the deletion.

N1/N2

The N1/N2 command adds a new record to a file. The user specifies the file to which the record is to be added together with the fields and field values to be used. Adabas performs all necessary modifications to the Associator and Data Storage.

If the N1 command is used, the ISN for the new record is assigned by Adabas. If N2 is used, the ISN is provided by the user.

Logical Transaction Control Commands (ET/BT)

An Adabas logical transaction defines the logical start (BT) and end (ET) of the database operation being performed. If the user operation or Adabas itself terminates abnormally, these commands make it possible to restart a user, beginning with the last unsuccessfully processed transaction. ET/BT commands

- define the transaction start and end;
- restore pretransaction conditions if a situation occurs that prevents successful completion of the transaction; and
- read program-specified user data written during the transaction sequence.

Programs that use these commands are called ET logic programs. Although not required, Software AG recommends that you use ET logic.

This section covers the following logical transaction control commands:

- BT
- ET

BT

The BT command backs out the current transaction being processed.

All modifications resulting from updates performed during the transaction are removed, and all records placed in hold status during the transaction are released (unless kept in hold status by the multifetch option; see the section *Multifetch Operation Processing*).

ET

The ET command indicates the end of a logical transaction.

An ET command causes Adabas to physically store all data protection information related to the transaction. This information is used to apply all the updates performed during the transaction at the start of the next Adabas session if the current session is terminated before these updates are physically applied to the database.

The ET command releases all the records that have been placed in hold status during the transaction (unless kept in hold status by the multifetch option; see the section *Multifetch Operation Processing*). The ET command may also be used to store user data in an Adabas system file. This data may be retrieved with an OP or RE command.

Special Commands

Special commands perform many of the "housekeeping" functions required for maintaining the Adabas database environment. Commands in this group

- open and close a session (but do not control a transaction);
- write data protection information and checkpoints; and
- set and release record hold status.

This section covers the following special commands:

- CL
- C1
- C3

- C5
- HI
- OP
- RC
- RE
- RI

CL

The CL command terminates a user session, releasing all records held for that user. It

- physically writes all current data protection information to the data protection log;
- releases all records currently in hold status for the user;
- releases all the command IDs and corresponding ISN lists currently assigned to the user; and
- stores user data in an Adabas system file (optional).

C1

The C1 command causes a checkpoint to be taken.

The C1 command physically writes all current data protection information to the data protection log, and writes a checkpoint entry to the data protection log and the system checkpoint file. This checkpoint entry may be used as a reference point for subsequent removal or reapplication of updates. An option allows the C1 command to initiate a buffer flush.

C3

The C3 command, issued only by exclusive control/update users (who are not using ET logic), writes a SYNX-03 checkpoint in the Adabas checkpoint file. This checkpoint

- contains the current data protection log and block number.
- may be used to restore the database (or certain files) to the status in effect at the time the checkpoint was taken. This may be necessary before a program performing exclusive control updating can be rerun or restarted.

If command option 2 is specified, the C3 command also stores user data in the Adabas checkpoint file for restart purposes. The stored data may be subsequently read with an OP or RE command.

C5

The C5 command writes user data to the Adabas data protection log.

The data can be read subsequently using the ADASEL utility.

HI

The HI command places a record in hold status. The user specifies the file and ISN of the record to be placed in hold status.

A record placed in hold status cannot be updated by another user until it is released.

OP

An OP command is mandatory when any of the following apply:

- the nucleus is run with ADARUN parameter OPENRQ=YES
- exclusive file control (EXF) is to be performed
- user data that was stored in an Adabas system file by a previous ET command is to be read
- user data is to be stored in an Adabas system file, using a C3, CL, or ET command
- the user is to be assigned a special processing priority
- the user is to be an access-only user (no update commands permitted).
- a transaction time limit and/or a non-activity time limit is to be set for the user that differs from that specified by ADARUN parameters TT and/or TNAx, respectively. The setting for a user must conform to the maximum setting set by the ADARUN parameters MXTT and MXTNA, respectively.
- special data encoding and/or architecture is to be specified for the user session.

It may also be used to set the maximum number of records that the user can place in hold status at the any given time, and the maximum number of command IDs the user may have active at the same time.

RC

The RC command may be used to release one or more command IDs currently assigned to a user, or to delete one or all global format IDs.

RE

The RE command reads user data previously stored in an Adabas system file by CL or ET commands.

RI

The RI command releases a record from hold status.

The user specifies the file and ISN of the record to be released. The user may also request that all records currently held by the user are to be released. The RI command should be issued by non-ET users only.

Transactions and ET Logic

This section describes the concept of a "transaction" and an ET logic user; it explains ET logic operations including normal and abnormal transaction termination processing and the storage and retrieval of user (ET) data.

This section covers the following topics:

- What is a Logical Transaction?
- Transaction Sequence Number
- ET Transaction Time Limit
- Back Out Transaction (BT) Command
- Autobackout
- End Transaction (ET) Command
- User (ET) Data
- Adabas User ID

What is a Logical Transaction?

A "logical transaction" is the smallest unit of work (as defined by the user) that must be performed in its entirety to ensure the logical consistency of the information in the database. Users who use logical transaction commands are referred to as "ET logic users".

A logical transaction comprises one or more Adabas commands that read/update the database as required to complete a logical unit of work. A logical transaction begins with the first command that places a record in hold status and ends when an ET, BT, CL, or OP command for the same user is issued.

The RE (read ET data) command can be used to retrieve user restart data stored by the ET or CL command.

Transaction Sequence Number

When a program issues an ET or CL command, Adabas returns a transaction sequence number in the command ID field of the ET or CL command’s control block. The transaction sequence number is a count of the total number of ET commands issued thus far during the user session.

The transaction sequence number is set to 1 for the first ET command issued by the user. The first ET command following the OP command returns transaction sequence number 2 in the command ID field. Each subsequent OP command returns the transaction sequence number of the last ET command issued by that same user.

ET Transaction Time Limit

Adabas provides a transaction time limit for programs that use ET logic. The time measurement for a transaction begins when the first command places a record in hold status, and ends when the program issues an ET, BT, or CL command.

The time limit is set with the ADARUN TT parameter; a transaction time limit that overrides this general limit for a specific user can be set with the OP command; this limit is controlled by the ADARUN MXTT parameter.

If a transaction exceeds the prescribed limit, Adabas generates a BT (back out transaction) command. The BT command removes all the updates performed during the transaction and releases all held records.

Adabas returns response code 9 for the next command issued by the user indicating that the current transaction has been backed out. The user can either repeat the backed out transaction from the beginning or begin another transaction.

Back Out Transaction (BT) Command

The BT command removes all updates made during the transaction currently being processed. This may be necessary because of a program error, a timeout, or when Adabas determines that the transaction cannot be completed successfully.

A BT command also performs an implied ET command, which releases all the records held during the transaction unless otherwise specified by the multifetch option; for more information, see the section *Multifetch Operation Processing*.

For example, the command sequence

```
FIND (S4)
  UPDATE (A1) (modify field XX to value 20)
  FIND (S4)
  UPDATE (A1) (modify field YY to value 50)
  END TRANSACTION (ET)
  FIND (S4)
  UPDATE (A1) (modify field XX to value 10)
  BACKOUT TRANSACTION (BT)
```

results in the field values of XX = 20 and YY = 50. The second update to field XX is removed by the BT command.

Autobackout

Autobackout, which is performed only for ET logic users, backs out transactions automatically in sessions that end abnormally. Adabas performs an internal BT (back out transaction) followed by autorestart, and then returns response code 9 to indicate that the last transaction has been backed out.

Autobackout occurs at the end of any nucleus session that was terminated with HALT.

Autobackout also occurs at the beginning of the next Adabas session to remove any updates that were performed in partially completed transactions by ET logic users during the previous terminated session.

When response code 9 indicates that the transaction was backed out, the user has the option of either reissuing the transaction from the start or beginning another transaction.

To restart the backed-out transaction, a terminal operator may need to reenter the data for the transaction, or an internal restart may have to be performed from the beginning of the "update phase" of the transaction; the "update phase" of a transaction begins with the first command that places a record in hold status.

End Transaction (ET) Command

The ET command must be issued at the end of each logical transaction. Successful execution of an ET command ensures that all the updates performed during the transaction will be physically applied to the database regardless of subsequent user or Adabas session interruption.

Updates performed within transactions for which ET commands have not been successfully executed will be backed out by the autobackout routine (see *Autobackout*).

Unless otherwise specified by the multifetch option, the ET command releases all records held by the user during the transaction. Adabas returns a unique transaction sequence number (see *Transaction Sequence Number*) that can be used to identify the transaction for auditing or restart purposes.

The ET command may also be used to store user data in an Adabas system file. This data may be used for user restart purposes, and may be read with an OP or RE command.

User Program	Adabas
FIND (S4), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
FIND (S1), HOLD ISN (HI), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
END TRANSACTION (ET)	data protection information for the transaction is written to the Adabas Work and data protection log, ending the transaction
FIND (S4), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
FIND (S1), HOLD ISN (HI), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
. . . Adabas or user session interruption . . .	

When the next Adabas session is started, or when the user is timed out, both updates for transaction 1 are physically written to the database (if they were not previously written). The updates for transaction 2 are not physically written (or will be backed out) because no ET command was processed for this transaction.

User (ET) Data

User data (ET data) may be stored with an ET or CL command and read with an OP or RE (read ET data) command.

One record of user data is kept for each user ID. The data is maintained until the next ET or CL command is issued in which user data is provided.

Each user data record is also written to the Adabas data protection log with each checkpoint written by the transaction. This data may be subsequently read with the ADASEL utility.

User data is primarily used to

- store data needed to restart an operation (e.g., input message data, transaction identification data, transaction summary data);
- store intermediate data to be used by subsequent transactions (e.g., for audit trail purposes);
- communicate with other users. The data stored may be read by other users provided that the ID of the user who stored the data is supplied in the OP command.

Software AG also suggests using the user data facility to

- establish an installation standard for the user data record format and store data for each update transaction with the ET command;
- read the user data and display it in a standard message format when the user logs on to an application. The user is thus informed of the last successful updating activity which corresponds to his user ID.

Software AG recommends that you keep ET commands that provide user data separate from those that do not. Combining the two types of ET command may cause significant additional overhead by forcing Adabas to repeatedly copy user data from the Adabas Work dataset (where it is temporarily stored) to the protection log.

Adabas User ID

The user ID is an identifier used by Adabas to store and retrieve user (ET) data and to assign a special processing priority to a user. The user ID is specified in the additions 1 field of the OP command.

A user ID provided by the user must

- begin with the character A (X’C1’) through 9 (X’F9’); and
- be unique to ensure that the user (ET) data is related to the user regardless of the terminal used. User (ET) data is maintained until the user’s next ET or CL command in which user (ET) data is provided.

If the user either does not provide a user ID or provides an invalid user ID, Adabas establishes a default user ID for the user session, and any user (ET) data stored by the user is deleted when the current session is terminated.

To avoid later limitations, Software AG recommends that you always specify a user ID.

Competitive Updating

Competitive updating is in effect when two or more users (in multiuser mode) are updating the same Adabas file(s).

This section describes the Adabas facilities used to ensure data integrity in a competitive updating environment. These include record hold/release and the avoidance of resource deadlock, as well as exclusive control updating.

This section covers the following topics:

- ET User Record Hold
- Avoiding Resource Deadlock
- Exclusive Control Updating

ET User Record Hold

The record hold facility allows the ET user to place a record in hold status for updating without allowing interim updating of the record by another user. Adabas "holds" a record by placing the ISN of the record in the hold queue; as a result, record hold is also called "ISN hold".

Record hold is applicable for all ET logic users; Adabas does not place an ISN in hold status for EXU (exclusive control updating) users. See the section *Exclusive Control Updating*.

This section covers the following topics:

- Record Hold Commands
- Record Update Using the Hold Option
- Record Release

Record Hold Commands

A record is held by using the "find with hold" command (S4), "read with hold" commands (L4, L5, L6), an A1 or E1 command in which the hold option is specified, or a "hold record" (HI) command. An N1/N2 command issued by an ET logic user also places the added record in hold status.

The successful completion of any of these commands places the record (ISN) in hold status. If the record is already being held by another user, the user issuing the record hold command is placed in a wait status until the record becomes available, at which time Adabas reactivates the command.

If the R (return code) or O (multifetch/return code) option is used with any of the record hold commands and the record to be held is already being held by another user, Adabas returns response code 145 instead of placing the user in a wait status.

A user who issues a "find" (S1) or "read" (L1, L2, L3) command is able to access the record regardless of the fact that the record is in hold status for another user.

Record Update Using the Hold Option

Users can update or delete any records they have in hold status by issuing an A1 or E1 command.

An A1 command is executed only if the record is either in hold status for the requesting user, or free from hold status and the user specifies that the record be held. If the record is currently held by another user, the requesting user is either placed in "wait" status or, if the user requests, receives response code 145. If the record is not in hold status, response code 144 is returned, indicating that the record was placed in hold status.

If an E1 command (without the hold option) is issued for a record that is not in hold status for the user, Adabas places the record in hold status for the user provided that the record is not in hold status for another user. If a user does not place a record to be deleted in hold status, there is no guarantee that the record will not be updated or deleted by another user before the E1 command is executed.

If an N1/N2 command is issued and there is no available space in the hold queue, response code 145 is returned.

Record Release

An ET logic user releases records from hold status with the ET command following each logical transaction, and with a "close" (CL) command at the end of the Adabas session. Programs using ET logic should not release records with the RI command if any updating has been performed during the current transaction, since this could result in a loss of data integrity.

For example, a record is updated and then released with an RI command by ET user 1, and the transaction continues. In the meantime, the same record is updated by user 2, who then ends the session with an ET command. If user 1’s transaction is subsequently backed out due to an error or transaction timeout, the updates performed by both users 1 and 2 are removed even though user 2’s transaction completed successfully with an ET command.

Therefore, user programs that employ ET logic should only release records at the completion of a logical transaction with the ET command. Non-ET logic users should use A1 or E1 commands to update or delete records and then release them.

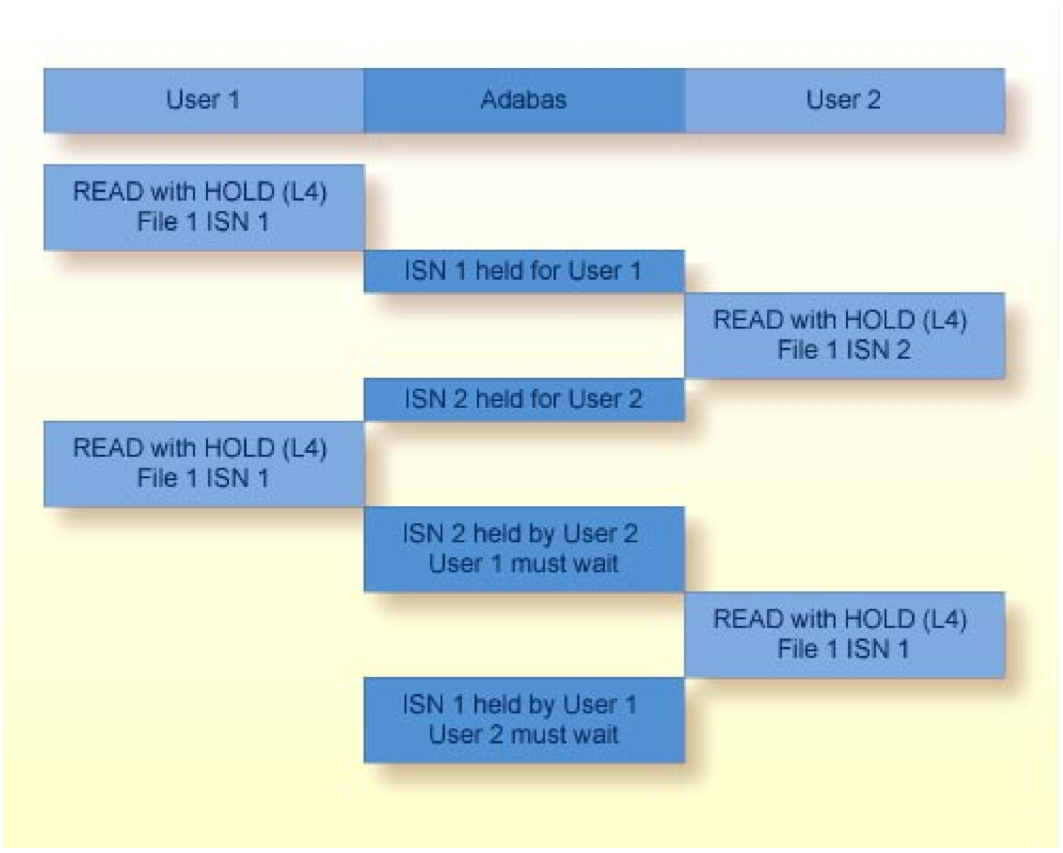
The multifetch option allows records (and their ISNs) to be released from hold status selectively if a non-zero count and the file number/ISN is specified in the ISN buffer when the ET (or BT) command is issued. See the section *Multifetch Operation Processing*.

The CL command releases all records in hold status for the issuing user, whether an ET user or not.

Avoiding Resource Deadlock

A resource deadlock would occur if two users were placed in wait status because each had requested a record that was currently in hold status for the other user.

For example:


User 1
Adabas
User 2
READ with HOLD (L4) File 1 ISN 1
ISN 1 held for User 1
READ with HOLD (L4) File 1 ISN 2
ISN 2 held for User 2
READ with HOLD (L4) File 1 ISN 1
ISN 2 held by User 2 User 1 must wait
READ with HOLD (L4) File 1 ISN 1
ISN 1 held by User 1 User 2 must wait

Resource Deadlock Example

Adabas detects the potential deadlock prior to putting user 2 in wait state and returns a response code 145 to the second user.

Exclusive Control Updating

A user may request exclusive control of one or more Adabas file(s) to prevent other users from updating the file during the execution of the user session.

Exclusive control is requested with the OP command and is given only if the file for which exclusive control is requested is not already opened for update by another user or utility. If the requested file is not available, the OP command returns response code 48.

In addition to preventing competitive updating of a file, exclusive control may be used to simplify recovery procedures, in that the file(s) may be restored regardless of other user activity.

The record hold commands need not (but may) be used for files for which the user has exclusive control. Adabas disables hold logic processing for files being updated under exclusive file control.

Exclusive control users are assumed to be non-ET logic users, and therefore not controlled by the ET timeout restrictions; however, if a non-ET user issues an ET command, that user automatically becomes an ET logic user and is subject to transaction timeout restrictions if records are put in "hold" status for any reason. That user retains exclusive control until either finished or timed out.

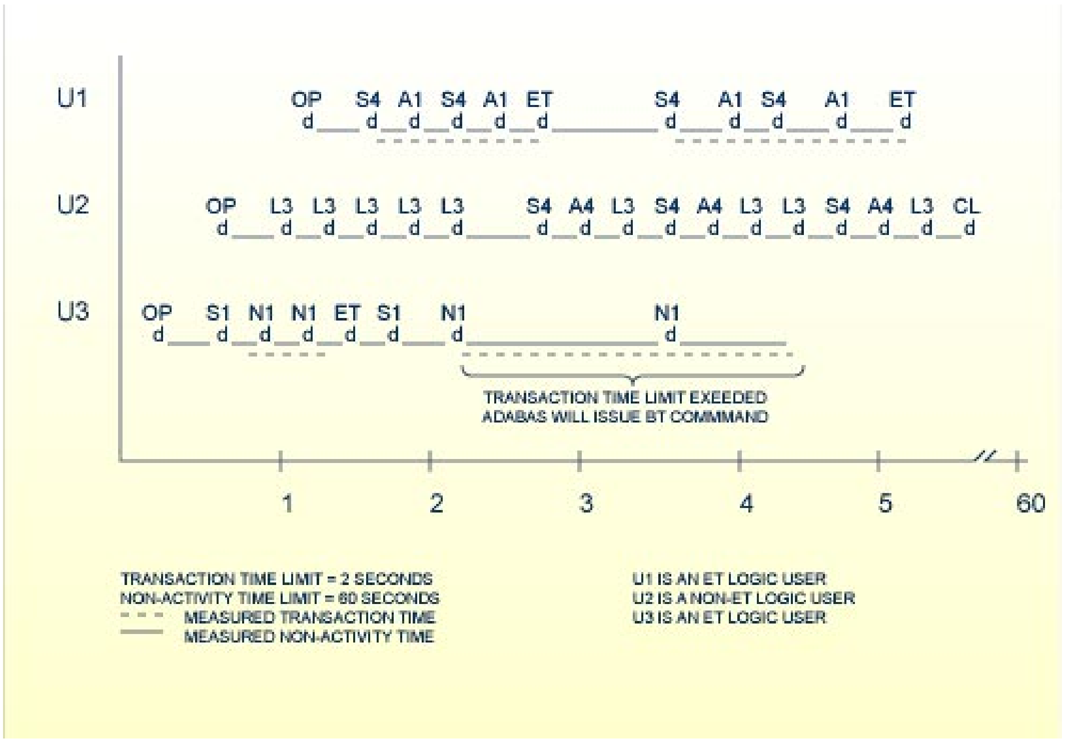
Users performing exclusive control updating can use the C1 command to request that a checkpoint be taken. The C1 checkpoint acts as a reference point to either remove updates that have been applied after the checkpoint, or to reapply updates that were applied before the checkpoint.

Non-Activity Time Limit

All users (access-only, ET logic, or exclusive control) are subject to a non-activity time limit. Different non-activity time limits may be defined for each user type with the ADARUN TNAA, TNAE, and TNAX parameters (see the *Adabas Operations* documentation for more information).

A user-specific non-activity time limit may also be set with the OP command; the maximum is controlled with the ADARUN MXTNA parameter. For more information, see the *OP Command* description.

The action taken by Adabas when a user exceeds the non-activity time limit is as follows:


U1
OP S4 A1 S4 A1 ET S4 A1 S4 A1 ET
U2
OP L3 L3 L3 L3 L3 S4 A4 L3 S4 A4 L3 L3 S4 A4 L3 CL
U3
OP S1 N1 N1 ET S1 N1 N1
TRANSACTION TIME LIMIT EXEEDED
ADABAS WILL ISSUE BT COMMAND
1 2 3 4 5 60
TRANSACTION TIME LIMIT = 2 SECONDS
NON-ACTIVITY TIME LIMIT = 60 SECONDS
MEASURED TRANSACTION TIME
MEASURED NON-ACTIVITY TIME
U1 IS AN ET LOGIC USER
U2 IS A NON-ET LOGIC USER
U3 IS AN ET LOGIC USER

Transaction and Non-Activity Time Limits

This section covers the following topics:

- For an ET Logic User
- For an Exclusive File Control User (EXF)
- For an Access Only User

For an ET Logic User

For an ET logic user, Adabas

- issues a BT command for the current user transaction (if necessary);
- releases all records held during the transaction;
- deletes the user’s file list in the UQE; and
- deletes all command IDs for the user.

Adabas returns response code 9 to the user when the next command is issued if

- the user was not in ET status when the timeout occurred; or
- the user was in ET status when the timeout occurred and provided a non-blank user ID in the OP command.

If the user was at ET status when the timeout occurred and did *not* provide a non-blank user ID in the OP command, the user’s UQE is deleted.

For an Exclusive File Control User (EXF)

For an exclusive file control user, Adabas deletes the user’s file list in the UQE. As a result, the user loses exclusive control of the file or files for which exclusive control was in effect.

In addition, all command IDs for the user are released and the user is changed to an access-only user.

For an Access Only User

For an access-only user, Adabas deletes the user’s file list in the UQE.

Adabas Calling Procedure

This chapter covers the following topics:

- Specifying an Adabas Call
- Syntax Conventions
- Control Block
- Format and Record Buffers
- Format Buffer Syntax
- Format Buffer Performance Considerations
- Record Buffer
- Format and Record Buffer Examples
- Search and Value Buffers
- Search Buffer Syntax
- Value Buffer
- Search/Value Buffer Examples
- ISN Buffer

Specifying an Adabas Call

This section describes the procedure used to call Adabas to execute an Adabas command; the syntax conventions used in the documentation; and the control block and buffers (format, record, search, value, and ISN) used to specify the Adabas command, and any additional information (parameters or operands) required for the command.

Adabas calls use the standard calling procedure provided by the host language (Assembler, COBOL, FORTRAN, or PL/I.

Note:
Examples of Adabas calls in each host language are provided with the programming examples in section *Programming Examples* of this documentation.

Each Adabas call must be accompanied by a parameter list. Each entry in the parameter list refers to a specific data area (buffer) defined in the user program. These buffers are used to transfer information to and from Adabas.

Parameter list entries must be provided in positional sequence. Parameters that are not used for a given command may be omitted provided that no required parameter follows; if it does, a dummy parameter must be provided.

The example below illustrates an Adabas call in COBOL.

```
CALL ’ADABAS’ USING parameter-list
```

where "parameter-list" is a positional list of user-defined fields in the control block and user-defined buffers specified in the following order:

control-block fields	command code command ID file number response code ISN
buffers	format (fields to be read, updated) record (values returned, updated) search (search criteria) value (search values) ISN (ISNs resulting from search)

The Adabas calling sequence may be illustrated as follows:

```
user program
                .
                .
CALL ’ADABAS’ USING parameter-list
                .
                .
CALL ’ADABAS’ USING parameter-list
                .
                .
```

Syntax Conventions

This section describes the syntax conventions common to all user-defined data areas (buffers). Notation specific to a particular buffer is introduced in the discussion of that area later in this section.

Variables and Constants

A *variable* is indicated in lowercase, non-bold type:

`name`

In the actual entry, a valid value must be supplied for the variable.

A *constant* is indicated in uppercase type:

`N`

You must enter a constant exactly as shown in the syntax statement. Symbols (except for brackets, braces, vertical bars, underlines, and three consecutive periods, which are notational devices only) are treated as constants and must also be entered exactly as shown in the syntax.

Optional Elements, Optional Values, and Default Values

Elements enclosed in *brackets* ([ ]) are optional:

```
name [i]
```

indicates that a value for "name" is required and may occur by itself or with "i".

Default values may be in effect for optional elements when no value is specified. Such values are *underlined* (__).

The set of options or possible values for an element is enclosed in brackets or braces:

- Within *brackets* ([ ]), one of the options or values can (but need not) be specified;
- Within *braces* ({ }), one of the options or values *must* be specified; there is no default.

Vertical bars (|) may be used to separate options or possible values listed horizontally within brackets or braces. Options or possible values may also be indicated by stacking them vertically within brackets or braces without the separating vertical bars.

Repeating Elements

Three consecutive periods indicate that an element can be repeated:

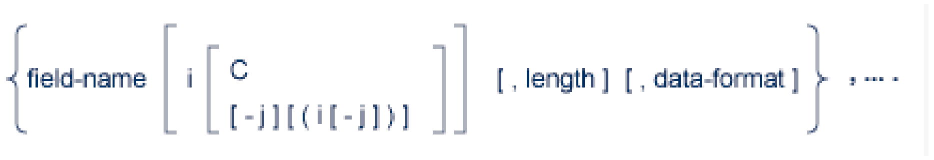
```
format,   ...
```

indicates that more than one format can be entered when separated from the previous one by a comma. When *braces* ({ }) precede the repetition indicator, they delimit the elements that must be specified once, but may, as a group, be repeated (up to a maximum number of times normally specified in the text). For example:

```
{, name1 , name2 }...
```

indicates that the group ",name 1, name 2" must be specified once but may be repeated.

Example:



Syntax Example

This example is a selected portion of the record format part of the format buffer syntax. It is used here only to illustrate common syntax conventions. (See *Format Buffer Syntax*, *record-format* for information specific to the format buffer.)

Following is information about reading the example syntax:

- The "field-name" and a final period (.) elements must be specified (required). All other elements in the syntax statement are optional.
- If specified, "i" is concatenated with the "field-name" since there is no intervening delimiter.
- If "i" is specified, then either "C" or the optional specifications stacked below it can be specified, but not both.
- The optional specifications below "C" allow the following combinations: i, i(i), i(i-j), i-j, i-j(i), i-j(i-j).
- If specified, "length" and "data-format" must each be preceded by a comma.
- The term enclosed in braces ({ }) and followed by a comma may be repeated any number of times.

Control Block

The control block and the related buffers specify which Adabas command is to be executed and provide any additional information (parameters or operands) required for the command. The name of the control block must always be the first operand specified in an Adabas call. The control block itself must have the format shown in the following *Adabas Control Block Definition* table.

Field	**Position Within Control Block**	**Length in Bytes**	**Format**
(see discussion)	1	1	binary
(reserved)	2	1	binary
COMMAND CODE	3-4	2	alphanumeric
COMMAND ID	5-8	4	alphanumeric / binary
FILE NUMBER	9-10	2	binary
RESPONSE CODE	11-12	2	binary
ISN	13-16	4	binary
ISN LOWER LIMIT	17-20	4	binary
ISN QUANTITY	21-24	4	binary
FORMAT BUFFER LENGTH	25-26	2	binary
RECORD BUFFER LENGTH	27-28	2	binary
SEARCH BUFFER LENGTH	29-30	2	binary
VALUE BUFFER LENGTH	31-32	2	binary
ISN BUFFER LENGTH	33-34	2	binary

Field	**Position Within Control Block**	**Length in Bytes**	**Format**
COMMAND OPTION 1	35	1	alphanumeric
COMMAND OPTION 2	36	1	alphanumeric
ADDITIONS 1	37-44	8	alphanumeric / binary
ADDITIONS 2	45-48	4	alphanumeric / binary
ADDITIONS 3	49-56	8	alphanumeric
ADDITIONS 4	57-64	8	alphanumeric
ADDITIONS 5	65-72	8	alphanumeric / binary
COMMAND TIME	73-76	4	binary
USER AREA	77-80	4	not applicable

The use of each applicable field in the control block is explained with each Adabas command description in this documentation. To ensure user program compatibility with later Adabas releases, all control block fields not used by a particular command should be set to zeros or blanks, depending on field type as shown in the *Adabas Control Block Definition* table.

The position of each field in the control block is fixed. For example, the command option 1 field must always be in position 35.

All values in the control block must be entered in the data type defined for the field. For example, the ISN field is defined as binary format; therefore, any entry made in this field must be in binary format.

Notes:

1. Adabas and other Software AG program products use some control block fields for internal purposes, and may return values in some fields that have no meaning to the user. These uses and values may be release-dependent, and are not appropriate for program use. Software AG therefore recommends that you use only the fields and values described in this documentation. *In addition, you should always initialize unused control block fields with either zeros or blanks, according to their field types.*
2. Some Adabas-dependent Software AG products return control block values such as response codes and subcodes. Refer to the documentation for those products for a description of the product-specific control block values.

Control Block Fields

The content of the control block fields and buffers must be set before an Adabas command (call) is issued. Adabas also returns one or more values or codes in certain fields and buffers after each command is executed.

The following descriptions of the Adabas control block fields are valid for most Adabas commands; however, some Adabas commands use some control block fields for purposes other than those described here.

This section covers the following topics:

- Byte 1
- Command Code
- Command ID
- File Number
- Response Code
- ISN
- ISN Lower Limit
- ISN Quantity
- Buffer Length: Format, Record, Search, Value, and ISN
- Command Options 1/2
- Additions 1
- Additions 2
- Additions 3
- Additions 4
- Additions 5
- Command Time
- User Area

Byte 1

The first byte of the Adabas control block (ADACB) is used by the Adabas API to determine the processing to be performed. See *Linking Applications to Adabas* in the *Adabas Operations* documentation for more information.

The values for logical requests are:

Hex	**Indicates ...**
X’00’	a 1-byte file number (file numbers between 1 and 255) or DBID.
X’30’	a 2-byte file number (file numbers between 1 and 65535) or DBID.
X’40’	values greater than or equal to a blank. These are accepted as "logical application calls" to maintain compatibility with earlier releases of Adabas.

Note:
The X’44’, X’48’, and X’4C’ calls are reserved for use by Software AG and are therefore *not* accepted.

All other values in the first byte of the ADACB are reserved for use by Software AG.

Because an application can reset the value in the first byte of the ADACB on each call, it is possible to mix both one- and two-byte file number (DBID) requests in a single application. In this case, you must ensure the proper construction of the file number (ACBFNR) and response code (ACBRESP) fields in the ADACB for each call type. See the discussions of these fields for more information.

Software AG recommends that an application written to use two-byte file numbers always place X’30’ in the first byte of the ADACB, the logical database ID in the ACBRESP field, and the file number in the ACBFNR field. The application can then treat both the database ID and file number as 2-byte binary integers, regardless of the value for the file number in use.

Command Code

The command code defines the command to be executed, and comprises two alphanumeric characters (for example, OP, A1, BT).

Command ID

The command ID field is used by virtually all Adabas commands to identify users, their transactions, and decoded formats for reuse by subsequent instructions. The alphanumeric command ID is either "user-specified" or "system-generated", and can be either "local" for a given user’s internal formats or "global", allowing many users to access the decoded formats. See the section *General Programming Considerations* for more information about command IDs. For ET, CL, and some OP commands, Adabas returns a binary transaction sequence number in the command ID field.

File Number

Note:
For commands that operate on a coupled file pair, this field specifies the primary file from which ISNs or data are returned.

The file number may be one or two bytes.

Single-byte File Numbers and DBIDs

For an application program issuing Adabas commands for file numbers between 1 and 255 (single byte), build the control block as follows:

Position	Action
1	Place X’00’ in the first byte of the ADACB.
9	Place the file number in the second (rightmost) byte of the ACBFNR field of the ADACB. The first (leftmost) byte of the ACBFNR field is used to store the logical (database) ID or number.

If the first byte in ACBFNR is set to zero (B’0000 0000’), the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data, or the default database ID value assembled into the link routine at offset X’80’. Applications written in Software AG’s Natural language need not include the first byte of the ADACB because Natural supplies appropriate values.

Double-byte File Numbers and DBIDs

Adabas permits the use of file numbers greater than 255 on logical requests. For an application program issuing Adabas commands for file numbers between 256 and 5000 (two-byte), build the control block as follows:

Position	Action
1	Place X’30’ in the first byte of the ADACB.
9	Use both bytes in ACBFNR for the file number, and use the two bytes in ACBRESP for the database (logical) ID.

If the ACBRESP field is zero, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data, or the default database ID value assembled into the link routine at offset X’80’.

Response Code

The response code field is used for two-byte database IDs.

It is also always set to a value when the Adabas command is completed. Successful completion is normally indicated by a response code of zero. For repeatable commands that process sequences of records or ISNs, other response codes indicate end-of-file or end-of-ISN-list. Non-zero response codes are defined in the *Adabas Messages and Codes*.

ISN

The ISN field both specifies a required four-byte Adabas ISN value required by the command and, where appropriate, returns either the first ISN of a command-generated ISN list, or an ISN of the record read by the command.

ISN Lower Limit

ISN lower limit specifies the starting point in an ISN list or range where processing is to begin. For OP commands, an optional user-specific non-activity timeout value can be specified in this field. An OP command also returns Adabas release information in this field (see also the additions 5 field description). When using the multifetch option, this field holds an optional maximum count of prefetched records to return; if zero, there is no limit.

ISN Quantity

The ISN quantity is a count of ISNs returned by a command. The count can be a total of all ISNs in an ISN list, or the total ISNs entered into the ISN buffer from a larger pool of ISNs by this operation. The OP command uses this field to specify an optional user-specific transaction time limit; it returns system and call type information flags in the ISN quantity field (see also the additions 5 field description). In addition, Sx commands using security-by-value set this field to 1 when *more than one* ISN meet the search criteria.

Buffer Length: Format, Record, Search, Value, and ISN

The format, record, search, value, and ISN buffer length fields specify the size of the related buffers. A buffer’s size usually remains the same throughout a transaction. In some ISN-related operations, the ISN buffer size value determines how a command processes ISNs; for example, specifying a zero ISN buffer length causes some commands to store a resulting ISN list in the Adabas work area. If a buffer is not needed for an Adabas command, the corresponding length value should be set to zero. In some cases (multifetch option, as an example), there is a limit on the length of the buffer; see the specific command descriptions for more information.

Command Options 1/2

The command option 1/2 fields allow you to specify processing options (ISN hold, command-level prefetching control, returning of ISNs, and so on).

Additions 1

The additions 1 field sometimes requires miscellaneous command-related parameters such as qualifying descriptors for creating ISN lists, or the second file number of a coupled file pair.

Additions 2

The additions 2 field returns compressed record length in the leftmost (high-order) two bytes and decompressed length of record buffer-selected fields in the rightmost (low-order) two bytes for all An, Ln, Nn, and S1/2/4 commands. OP (open) and RE (read ET data) commands return transaction sequence numbers in this field. If Entire Net-work is installed, some response codes return the node ID of the "problem" node in the leftmost two bytes of the additions 2 field.

If a command results in a nucleus response code, the addition 2 field’s low-order (rightmost) two bytes (47 and 48) can contain a hexadecimal subcode to identify the cause of the response code. For example, if no OP command began the session and the ADARUN statement specified OPENRQ=YES, a response code 9, subcode 66 is returned and these bytes are set to the hexadecimal value 0042 corresponding to the decimal 66. Response codes and their subcodes (as decimal equivalents) are described in the section *Nucleus Response Codes* in the *Adabas Messages and Codes*.

Additions 3

The additions 3 field is for providing a user‘s password for accessing password-protected files. If the file containing the field is actually password-protected, the password in this field is replaced with spaces (blanks) during command execution before Adabas returns control to the user program.

Note:
Some proprietary/optional features such as the Adabas External Security Interface (ADAESI) set this password, and it therefore should not be overwritten.

Additions 4

The additions 4 field must be set to a cipher code for those instructions that read or write encrypted (ciphered) database data files. For commands requiring multiple command IDs but no cipher code, one of the command IDs is specified in this field.

When processed by the nucleus, an Adabas call returns the Adabas release (version and revision) level numbers and the database ID in the low-order (rightmost) *three* bytes of the additions 4 field with the format "vrnnnn" where

v	is the Adabas version number;
r	is the Adabas revision level number; and
nnnn	is the number (hexadecimal) of the Adabas database that processed the call.

For example, " 741111" indicates that an Adabas version 7.4 nucleus on database 4369 processed the call.

There is one exception to this format: If the call is processed by an Adabas nucleus lower than version 5, the value returned is "404000".

Additions 5

The high-order (leftmost) two bits of the first byte of the additions 5 field control the unique or global format ID selection; the low-order (rightmost) four or eight bytes can contain either an optional unique or global format ID, respectively. Refer to the section *Using a Global Format ID* for a complete description of this feature. A global format ID to be deleted can be specified in this field for the RC (release command ID) command. When completed, the OP command returns any optionally specified non-activity and/or transaction timeout values in the additions 5 field.

Command Time

The command time field is used by Adabas to return the elapsed time which was needed by the nucleus to process the command. This does not include the times when the thread was waiting on Adabas I/O operations or other resources. The time, counted in 16-microsecond units, is called "Adabas thread time". The returned count is in binary format.

User Area

The user area field is reserved for use by the user program. When making logical user calls, the user area is neither written nor read by Adabas.

For compatibility with future Adabas releases, Software AG recommends that you set unused control block fields to null values corresponding to the field’s data type.

Programming Examples

Programming examples that show control block construction in each host language are provided in section *Programming Examples* of this documentation.

- *Examples for Assembler*
- *Examples for COBOL*
- *Examples for PL/ I*
- *Examples for FORTRAN*

Logging the Control Block

There are two formats for the command log. The newer format was first implemented in Adabas 5.2. The ADARUN parameter CLOGLAYOUT determines which format is used.

If CLOGLAYOUT=4, Adabas logs the following control block fields into the Adabas basic command logging area:

command code	command option 1
command ID	command option 2
file number	additions 2
response code	command time
ISN	

All other control block fields are logged into the extended command logging area.

If CLOGLAYOUT=5, all Adabas control block fields are logged in the basic command logging area. The extended logging area does not exist in this format.

The CLOGLAYOUT parameter is described in the *Adabas Operations* documentation. *Appendix A* in the *Adabas DBA Reference* documentation provides a detailed description of both command log formats.

Format and Record Buffers

The format buffer specifies the fields to be processed during the execution of an Adabas read/update command.

For read commands, the values of the fields specified in the format buffer are returned by Adabas in the record buffer.

Format Buffer	**`AA,BB`**	Name of the fields to be updated
Record Buffer	`value-AA value-BB`	Field values provided by user

For add/update commands, the new values for the fields specified in the format buffer are provided by the user in the record buffer.

Format Buffer	**`XX,YY`**	Name of the fields to be updated
Record Buffer	`value-XX value-YY`	Field values provided by user

For the OP command, the record buffer indicates the type of user and the files to be used.

The record buffer is also used for user data (OP, RE, CL, ET commands).

The format buffer must be long enough to hold the largest field definition contained in the related program, including the ending point (.).

Format Buffer Syntax

This section describes the syntax used to construct the format buffer.

Multiple record formats may be specified in the format buffer for files that contain various record formats. The specific record format to be used is determined by the value of a field in the file (for example, record type). When specifying multiple record formats, each one must be preceded by a format selection criterion.

The format buffer has the following syntax:

```
{ [ format-selection-criterion ] record-format } , ... .
```

A comma must be used to separate all format buffer entries. One or more spaces may be present between entries. The last entry may not be followed by a comma.

The format buffer must end with a period.

format-selection-criterion

The syntax of the format selection criterion is

```
( field-name operator { value , ... } )
```

field-name

- The field name used must be present in the FDT of the Adabas file being read.
- The field can be a multiple-value field.
- The field can be contained within a periodic group.
- NU or NC/NN option fields must have a non-null value; otherwise, the selection criterion is false.
- Group names, hyperdescriptors, and phonetic descriptors cannot be specified.

operator

EQ (or) =	equals
NE	not equal
LT (or) <	less than
GT (or) >	greater than
LE	less than or equal
GE	greater than or equal

value

The "value" is a numeric integer or an alphanumeric value. A series of values can be specified, separated by commas and using the EQ operator. An alphanumeric value must be enclosed within apostrophes (’value’).

Example:

```
(SA = 1) record-format-1, (SA = 2,3,4) record-format-2, (SA GE 5) record-format-3.
```

- If the value of field SA is 1, "record-format-1" is used;
- If the value of field SA is 2, 3 or 4, "record-format-2" is used;

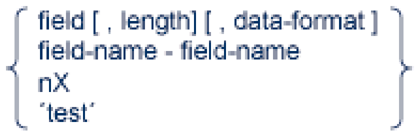
- If the value of field SA is equal to or greater than 5, "record-format-3" is used.

The first criterion that is met is used. If no criteria are met, a response code is returned.

record-format

The record-format is used to indicate which fields are to be read (read commands) or updated (update commands).

The syntax of the record format is



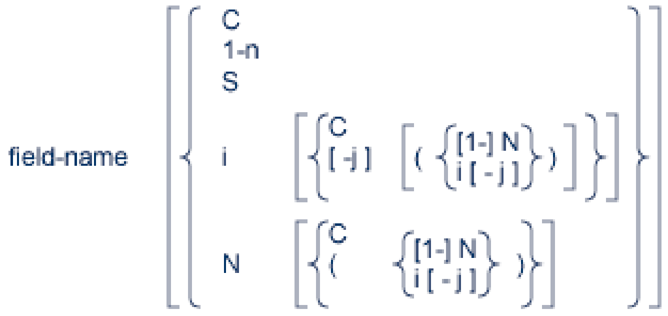
Record Format Syntax

The "field" term is explained in the following section. For information about the

- "length" and "data-format" terms, see page *Length and Data Format*.
- field series notation "field-name - field-name", see *Field Series Notation*.
- space notation "nX", see *Space Notation (nX)*.
- text insertion notation 'text', see *Text Insertion Notation*.

field

The syntax for "field" is



Field Syntax

This section discusses the "field-name" term. See the following additional information:

- *Count Indicator (C).*
- *Highest Occurrence/Value Indicator (N).*
- *SQL Significance Indicator (S).*

The "field-name" is the name of the field or group for which the value(s), range, or count is requested or for which a new value(s) is being provided. The name specified must be two characters in length and must be present in the FDT of the file being read/updated. The name can refer to an elementary, sub-/superfield, a multiple-value field, a group, or a periodic group.

A name that refers to a group results in all the fields within the group being referenced. Use of group names can greatly reduce the time required to process the command. A group name cannot be used if the group contains a multiple-value or variable-length field (no standard length).

For access commands, the same name may be specified more than once. In this case, the field value is returned multiple times.

For update commands, the same name cannot be used more than once (except in the case of multiple-value fields, as explained later in this section).

A sub-/superfield, or a sub-/superdescriptor name may be specified for access commands but not for update commands.

The following *Allowed Format Buffer Field Types* table illustrates the relationship between allowed single-value field type and type of command:

Cmd Type	Field	Subfield	Superfield	DE	SUBDE	SUPERDE	COLDE
Read (Ln) [1]	yes	yes	yes	yes	yes	yes	yes [3]
Add (Nn)	yes	no	no	yes	no	no	no
Find (Sn)	yes	yes	yes	yes	yes	yes	yes
Update (A1/4) [2]	yes	no	no	yes	no	no	no

Notes:

1. Format "C." may be used with read commands to request the record in its compressed format.
2. A field specified for an update command cannot
   be repeated;
   contain a "1-N"-type specification;
   specify a group and an element (field or group) contained in the group.
3. A collation descriptor (COLDE) can only be specified in the format buffer of the L9 command and only when the decode option has been specified in the user exit. The value returned is not the index value but the original field value.

For the Adabas file definitions used in all the examples in this section, see *File Definitions Used in Examples*.

Index or Range Notation

The following is the field name syntax for selecting multiple-value fields or occurrences of periodic groups:

```
field-name i [ - j ]    (periodic group or multiple-value field index or range)
```

where

i	is the periodic group or multiple-value occurrence
i-j	is the periodic group or multiple-value occurrence range

Periodic group names *must* be followed by a numeric or other appropriate suffix (see the discussions of the *Count Indicator (C)* and the highest occurrence/value indicator *Highest Occurrence/Value Indicator (N)* for more information). Specifying a periodic group name as simply "field-name" is incorrect syntax.

Multiple-value fields can be specified by explicitly identifying a particular value (indexing) or by referencing each value in sequence, letting Adabas assign an index based on the sequence. See *Multiple-Value Fields* for more information.

Example	**Selects . . .**
AA3	the third value of multiple-value field AA, the third occurrence of periodic group AA, or the first occurrence of the single- or multiple-value field AA contained within the third occurrence of a periodic group.
AA3-6	the third through sixth values of the multiple-value field AA, the third through sixth occurrences of periodic group AA, or the first occurrence of the (single or multiple-value) field AA contained within the third through sixth occurrences of a periodic group.

Periodic Groups

If a periodic group (or a field within a periodic group) is to be referenced, the specific occurrence to be used must be specified. This is accomplished by appending a one- to three-digit index (value 1-191, leading zeros are permitted) to the name.

An occurrence is identified by the name of the periodic group and the occurrence number.

Example	**Selects . . .**
GB3	the third occurrence of periodic group GB (fields BA3, BB3, BC3).

When selecting fields within one or more periodic groups, the field name and occurrences (values) are used; the group name is implied.

Example	**Selects . . .**
BB06	the sixth occurrence of field BB.

To refer to a range of occurrences of a periodic group (or a field within a periodic group), enter the periodic group or field name, followed by the first and last occurrence numbers separated by a hyphen. The occurrence numbers must occur in ascending sequence; a descending range is not permitted.

Example	Selects . . .
GB2-4	the second through fourth occurrences of periodic group GB, including all fields within these occurrences (BA2, BB2, BC2; BA3, BB3, BC3; and BA4, BB4, BC4).
BA2-4,BC2-4	the second through fourth occurrences of BA and BC (BA2, BA3, BA4, BC2, BC3, BC4).

Multiple-Value Fields

Multiple-value fields can be specified in the format buffer in two ways: by explicitly identifying a particular value (indexing) or by referencing each value in sequence, letting Adabas assign an index based on the sequence. These two methods apply as well to sub- or superdescriptors derived from multiple-value fields.

1. Indexing: To refer to a particular value of a multiple-value field, enter a one- to three-digit index (value 1-191, leading zeros permitted) after the field name.

Example	Selects . . .
MF2	the second value of the multiple-value field MF.
MF1, MF10	the first and tenth values of the multiple-value field MF.

To refer to a range of values for a multiple-value field, specify the first and last values desired, connected by a hyphen, immediately after field name. An ascending range must be specified.

Example	Selects . . .
MF1-3	the first three values of the multiple-value field MF.

2. Sequencing: To refer to multiple-value fields by repeating the field name, first specify the first value, then the second, and so on.

Example	Selects . . .
MF,MF	the first and second values of the multiple-value field MF.
AA,MF,AB,MF,AC,MF	the first three values of the multiple-value field MF and the values for the fields AA, AB and AC, in the order shown.

Both methods of referencing a multiple-value field can be used in the same format buffer. If no occurrence index is specified, an index that is one higher than the last index, proceeding from left to right, is implicitly (or explicitly) assigned. If the last index was specified as "N" or "1-N", referring to the highest existing occurrence, a new field specification without an explicit index will refer to the *same* field occurrence as the last specification. See *Highest Occurrence/Value Indicator (N)* for more information about the highest occurrence/value indicator N/1-N/NC.

For an update command where *all* format buffer references to a multiple-value field are specified without indexes (i.e., by sequencing), only those occurrences specified will remain in the record; all other occurrences that might exist are deleted. If *any one* reference to a multiple-value field is specified with an index, then only the specified occurrences are changed; all other occurrences remain unchanged.

Multiple-Value Fields within Periodic Groups

If a multiple-value field is within a periodic group, specifying the multiple-value field name implies the periodic group name. The group name, therefore, does not have to be specified; only the group occurrence or occurrence range is required. The general syntax for selecting a multiple-value field value or value range within a periodic group occurrence or occurrence range is

```
field-name i    [ - j  ]  ( i [ -  j ] )
```

where

i	is the periodic group occurrence
i-j	is the periodic group occurrence range
(i)	is the multiple-value field value
(i-j)	is the multiple-value field value range

To refer to a multiple-value field contained within a periodic group, enter the occurrence number of the periodic group after the field name, followed by the desired multiple-value field values or range of values enclosed within parentheses.

Example	**Selects . . .**
CB2(5)	the fifth value of the multiple-value field CB in the second occurrence of the periodic group that contains field CB.
CB2(1-5)	the first five values of the multiple-value field CB in the second occurrence of the periodic group that contains field CB.

To refer to either the same multiple-value field or the same ranges of multiple-value fields contained within a range of periodic group occurrences, enter the range of occurrences after the field name, followed by the multiple-value field value or range of values, enclosed within parentheses.

Example	**Selects . . .**
CB3-5(2)	the second value of multiple-value field CB in each of the third through the fifth occurrences of the periodic group containing field CB.
CB1-2(1-4)	the first four values of the multiple-value field CB in the first occurrence of the periodic group containing field CB, followed by the first four values of CB in the second occurrence of the periodic group.

Count Indicator (C)

To obtain the count of periodic group occurrences, or the count of existing values of a multiple-value field not in a periodic group, specify the periodic group or multiple-value field name followed by "C":

```
field-name C     count for multiple-value field or periodic group "field-name"
```

Example	Selects . . .
GBC	the highest occurrence number (also the occurrence count) in periodic group GB.
MFC	the number of existing values in multiple-value field MF that are not contained in a periodic group.

The count is returned in the record buffer as a one-byte binary number unless an explicit length and/or format is specified (see *Length and Data Format*).

To obtain the count of the existing values of a multiple-value field contained within a periodic group, enter the multiple-value field name followed by the group occurrence and "C":

```
field-name i C   count for multiple-value field "field-name" within a group occurrence
```

Example	Selects . . .
CB4C	the number of existing values of multiple-value field CB in the fourth occurrence of a periodic group.

For update commands, specifying "C" causes Adabas to skip in the record buffer the number of positions occupied by the count field, thus ignoring the count.

The user cannot directly update multiple-value or periodic group count fields. These count fields are updated by Adabas when multiple-value field values and periodic group occurrences are added or deleted.

Highest Occurrence/Value Indicator (N)

The indicator "N" selects the last value in a series of values comprising a multiple-value field, or the last occurrence of a periodic group, removing the need to know the number of the last value or occurrence.

The notation "1-N" selects all values comprising a multiple-value field, or all occurrences of a periodic group. For multiple-value fields in periodic groups, it is not possible to combine the specification 1-N for the group occurrence with any specification for the field occurrences.

The notation "NC" selects the count of the existing values of a multiple-value field in the last occurrence of the periodic group containing the field.

`field-name  N`	Last occurrence of periodic group "field-name" or the highest value of multiple-value field "field-name".
`field-name  NC`	Count of values for multiple-value field "field-name" in the last occurrence of the periodic group containing "field-name".
`field-name  NC`	Selects all values of multiple-value field "field-name" or all occurrences of the periodic group "field-name".
`field-name  N (N)`	The last value of multiple-value field "field-name" in the last occurrence of the periodic group containing "field-name".
`field-name  N(1-N)`	All values of multiple-value field "field-name" in the last occurrence of the periodic group containing "field-name".
`field-name  N( i [- j`	Value or range of values of multiple-value field "field-name" in the last occurrence of the periodic group containing "field-name".
`field-name i [- j ] (`	The last value of multiple-value field "field-name" in an occurrence or range of occurrences of the periodic group containing "field-name".
`field-name  i [-j ] (`	All values of multiple-value field "field-name" in an occurrence or range of occurrences of the periodic group containing "field-name".

Example	**Selects . . .**
MFN	the highest (last) value of multiple-value field MF. If multiple-value field MF contains four values, the fourth value (the last value entered) is selected.
GBN	the highest occurrence number of a periodic group GB.
MFNC	the count of existing values for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MF1-N	all values of the multiple-value field MF.
GB1-N	all occurrences of periodic group GB.
MFN(1-N)	all values for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MFN(N)	the highest value for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MFN(4)	the fourth value of the multiple-value field MF in the highest occurrence of the periodic group containing MF.
CB3-5(N)	the highest value of the multiple-value field CB in the third through fifth occurrences of the periodic group containing CB.
CB2(1-N)	all values of the multiple-value field CB in the second occurrence of the periodic group containing CB.

SQL Significance Indicator (S)

The "S" significance, or null, indicator and the corresponding null indicator value in the record buffer indicate whether a field’s value is significant, including zero or blank, or not significant (undefined). The S indicator can only be applied to elementary fields that are defined with the NC option, but not for an NU option field:

```
field-name S    SQL significance indicator
```

See the section *Record Buffer* for a description of the null value indicator, and the ADACMP utility description in the *Adabas Utilities* documentation for information about defining SQL null fields.

The related null indicator value in the record buffer, described below, has a two-byte standard length and fixed point format; this length and format cannot be overridden.

For update-type commands, a null value will be stored in the field if the corresponding null indicator value is X‘FFFF’ and no NN option is specified for the field. Otherwise, the null indicator must be X‘0000’. This, combined with the S indicator, shows that the field’s value given elsewhere in the record buffer is to be stored.

The S indicator is not required for update-type commands; if the S indicator is not specified, the field value is updated exactly as if the S indicator had been specified and the corresponding null indicator value had been set to zero (a null in the field is a value). See the examples below.

For read-type commands, the S indicator is required when the NC fields are defined without the NN option. If the S indicator is not present when a read command detects an NC-specified field and the field actually contains a null value, a response code 55 is returned.

Example:

This example uses the "field-name S" indicator with the two-byte null indicator in the record buffer to update or add a record, setting field AA equal to the SQL null value and ignoring the value for field AA:

Format Buffer	`AAS , AA , 2 , ... .`	Name of the fields to be read
Record Buffer	`FFFF  123F`	Field values returned by Adabas

For examples showing the use of the SQL significance indicator when a group or range of fields containing an NC field is specified, see the section *The SQL Significance Indicator and Field Series Notation*.

The "field-name S" indicator can be anywhere within the format buffer; that is, it need not precede the corresponding field element. For update-type commands, the format buffer cannot contain more than one element referring to the same field:

AAS.	valid for read/update commands
AA.	valid for read/update commands
AAS,AA,AAS.	valid for read commands only

This means that several format buffer elements referring to the same field cannot be specified for an update-type command:

AA,AA.	causes response code 44
AA,AAS,AA.	causes response code 44

Field Series Notation

The notation "field-name - field-name" may be used to refer to a series of consecutive fields (as ordered in an FDT). The user specifies the beginning and ending field names connected by a dash:

```
field-name - field-name  series notation
```

No multiple-value field or periodic group may be contained within the series.

A name that refers to a group may not be specified as the beginning or ending name, but a group may be embedded within the series.

Standard format and length is in effect for all the fields within the series. No length or format override is permitted.

Example	Selects . . .
AA-AC	the fields AA, AB, and AC.
AA-GC	nothing. The series may not contain a multiple-value field or a periodic group.
GA-AC	nothing. The series may not begin/end with a group.
AA,5,U,-AD	nothing. A length and/or format override is not permitted in a series notation.

The SQL Significance Indicator and Field Series Notation

When a group or range of fields contains a field specified with the NC option, the corresponding S operator is optional for read (Lx) commands. For update (A1) commands, the S operator must not be specified. Adabas assumes that the null indicator corresponding to the NC field in the format buffer is located just in front of the field’s value in the record buffer.

For example, given the following field definitions in the FDT:

```
01,GR
  02,AA,8,A
  02,BB,8,A,NC
  02,CC,8,A
```

if the format buffer of an update-type command specifies GR. or AA-CC. , the record buffer has the following structure:

```
AA-value null-indicator-BB BB-value CC-value
```

That is, the null indicator must be included in the record buffer sequence, although the S indicator was not (and must not be) specified in the format buffer.

If the format buffer of a read (Lx) command specifies GR,BBS. or AA-CC,BBS., the record buffer has the following structure:

```
AA-value null-indicator-BB BB-value CC-value
null-indicator-BB
```

In other words, the first appearance of the null indicator is implied in the record buffer while the second appearance was explicitly called for by the format buffer.

Length and Data Format

The length and format parameters are used if a field value is being provided or is to be returned in a length and/or format different from the standard defined for the field in the FDT. If the length and/or format parameters are omitted, the field value must be provided or is returned in the standard length and format of the field:

```
[ , length ] [ , data-format ]
```

Possible format/length conversions are suggested by the information in the following *Permitted Data Lengths and Formats* table. A format conversion cannot be specified for subfields or subdescriptors; superfields or superdescriptors; or hyperdescriptors.

Fmt	Max Length (in bytes)	Data Type	Compatible Formats
A	253	alphanumeric, left-justified	W
W	253 [1]	wide-character, left-justified	A
A,W	16,381 [1,2]	alphanumeric or wide-character with LA (long alpha) option; left-justified; preceded by optional two-byte binary (inclusive) length	W,A
B	126	binary; right-justified; unsigned	A,F,P,U
F	4	fixed-point; right-justified; signed; two or four bytes	A,B,P,U
G	8	floating-point; four or eight bytes	none
P	15	packed decimal; signed; positive=A,C,E, or F; negative=B or D	A,B,F,U
U	29	unpacked decimal; signed; positive=A,C,E, or F; negative=B or D	A,B,F,P

Notes:

1. Like an alphanumeric field, a wide-character field may be a standard length in bytes defined in the FDT, or variable length. Any non-variable format override for a wide-character field must be compatible with the user encoding; for example, a user encoding in Unicode requires an even length (max. 252 bytes).

2. Maximum long alpha length if the length (variable field length notation) precedes the field in the record buffer; otherwise, the maximum length is 253 bytes.

The length specified must be large enough to contain the value in the chosen format, but cannot exceed the maximum length permitted.

If a length of zero is specified, or if "field-name" refers to a variable-length field (no standard length), the value returned by Adabas in the record buffer is preceded by a one-byte binary field containing the length of the value (including the length byte itself). For update commands, you must provide this length byte at the beginning of the record buffer.

The format specified must be compatible with the standard format of the field.

- Conversion between packed/unpacked decimal values and binary is limited to values between 0 and 2,147,483,647.
- Conversion from a numeric format to alphanumeric results in an unpacked value, left justified, without leading zeros and with trailing blanks. For example, the three-byte packed value "10043F" would be converted to "F1F0F0F4F3404040". Value truncation is possible with this type of conversion.

Edit Mask Notation (Read Operations Only)

Edit masks are used according to the standard edit mask rules used in the COBOL programming language.

An edit mask may only be specified for numeric fields. All data returned by Adabas to an edited field is converted to unpacked decimal format regardless of the standard format of the field. A maximum of 15 digits (not counting edit characters) can be returned to an edited field.

For a field with an edit format specified, the length parameter must be large enough to contain the field value plus all required edit characters.

Format	Generates the edit mask . . .
E1	zzzzzzzzzzzzzzz
E2	zzzzzzzzzzzzz9-
E3	zzzzzzzzz99.99.99
E4	zzzzzzzzz99/99/99
E5	z.zzz.zzz.zzz.zzz,zz
E6	z,zzz,zzz,zzz,zzz.zz
E7	z,zzz,zzz,zzz,zz9.99-
E8	z.zzz.zzz.zzz.zz9,99-
E9	*,***,***,***,**9.99-
E10	*.***.***.***.**9,99-
E11	user-designated mask
E12	user-designated mask
E13	user-designated mask
E14	user-designated mask
E15	user-designated mask

Note:
Although edit formats E3 and E4 provide space for the century digits (see the following examples), they do not *enforce* date formats that are compatible with year 2000 requirements.

Examples:

Format Buffer	Field Value	Edited Value
XC,15,E1.	009877	bbbbbbbbbbb9877
XC,8,E4.	301177	30/11/77
XB,5,E7.	-366	3.66-
XB,7,E9.	542	**5.42
Y2,10,E4.	20000229	2000/02/29

Space Notation (nX)

The nX syntax is used differently for read and update commands:

nX

For read commands, nX indicates that "n" spaces are to be inserted in the record buffer by Adabas immediately before the next field value:

Format Buffer	`AA , 5X , BB`	Name of the fields to be read
Record Buffer	`value-AA 5-blanks value-BB`	Field values returned by Adabas

For update commands, nX causes "n" positions in the record buffer to be ignored by Adabas:

Format Buffer	`XX , 5X , YY`	Name of the fields to be updated
Record Buffer	`value-XX ignore-5-bytes value`	Field values provided by user

Text Insertion Notation

The ’text’ syntax is used differently for read and update commands:

`’text’`

For read commands, the character string specified in the format buffer is to be inserted in the record buffer immediately before the next field value. The character string provided can be 1-255 bytes long, and may contain any alphanumeric character except a quotation mark.

For example:

Format Buffer	`AA , ’text’ , BB`	Name of the fields to be read
Record Buffer	`value-AA text value-`	Field values returned by Adabas

For update commands, the number of positions enclosed within the apostrophes in the format buffer will be ignored in the corresponding positions of the record buffer.

Format Buffer Performance Considerations

Performance improvements may be achieved by using the following guidelines during format buffer construction:

- Use group names wherever possible rather than referring to elementary fields individually. Use of group names reduces the time required by Adabas to interpret the format buffer.
- Use of the field series notation does not result in performance improvements. A field series notation is converted by Adabas into a series of elementary fields.
- Use length and format overrides only when necessary. Using overrides requires additional processing time when interpreting the format buffer and when processing the field.
- If the same fields of a record are to be read and then updated, the same format buffer should be used for the read and update commands. For more information, refer to the descriptions of command and format IDs beginning on page .

- You should request periodic (PE) group and multiple-value (MU) field occurrences, based on the requirements of the application. In other words, do not arbitrarily request all occurrences; doing so requires extra time to translate the format buffer, and may mean decompressing numerous empty occurrences. Generally, the AA1-N field argument is the most efficient for selecting periodic group occurrences.

Record Buffer

The record buffer is used primarily with read, search, and update commands.

For read commands, Adabas returns the requested field values in this buffer in the order specified by the format buffer. Each value is returned in the standard length and format defined for the field unless a length and/or format override was specified in the format buffer. If the value is a null value, it is returned in the format that is in effect for the field, as follows:

Field Type	Null value represented by . . .
Alphanumeric (A)	blanks (hex ’40’) or blank of user override encoding
Binary (B)	binary zeros (hex ’00’)
Fixed (F)	binary zeros (hex ’00’)
Floating Point (G)	binary zeros (hex ’00’)
Packed (P)	decimal packed zeros with sign (hex ’00’ followed by ’0A’, ’0B’, ’0C’, ’0D’ or ’0F’ in the rightmost, low-order byte)
Unpacked (U)	decimal unpacked zeros with sign (hex ’F0’ followed by ’C0’ or ’D0’ in the rightmost, low-order byte)
Wide-character (W)	Unicode blanks (hex ’20’) or blank of user override encoding

Note:
SQL-compatible null values in NC/NN option fields require the additional null value and significance indicator. See *Specifying and Reading the SQL Null Indicator*, and *SQL Significance Indicator (S)*.

Adabas returns the number of bytes equal to the combined lengths (standard or overridden) of all requested fields.

When updating a record, you must specify the new value in the record buffer. If a null value is being provided, it must be provided according to the field type in effect, as described above.

The record buffer is also used to transfer information between the user program and Adabas in the following commands:

Command	Data Provided	Data Returned
OP	Files to update and the operation type (ET, exclusive control)	User data (optional)
LF	-	Field definitions for the file
RE	-	User data stored in system file
C5	Protection log user data	-
ET/CL	User data (optional)	-

Specifying and Reading the SQL Null Indicator

To support Software AG’s Adabas SQL Server (ESQ) and other structured query languages (SQLs), fields defined with the NC/NN (not-counted/null-not-allowed) options indicate an SQL-significant null with a two-byte binary null indicator in the record buffer.

Whether a field’s "zero" value is significant or an irrelevant null (unspecified) depends on the null indicator specified in the record buffer when the value is entered or changed, or returned in the record buffer when the value is read.

In addition to specifying or reading the value itself, either

- set the null indicator into the record buffer position that corresponds to the field’s designation in the format buffer for an update operation, or
- ensure that your program examines the null indicator (if any) returned in the record buffer position corresponding to the field’s position in the format buffer for a read operation.

The null indicator is always two bytes long and has fixed-point format, regardless of the data format.

For a read (Lx) or find with read (Sx with format buffer entry) command, the null indicator value returns one of the following (hexadecimal) null indicator values, according to the actual value that the selected field contains

FFFF	a null value in this field is not significant.
0000	a null value in this field is a significant value; that is, a true zero or blank.
xxxx	the field was truncated. The null indicator contains the length (xxxx) of the entire value as stored in the database record.

For an update (Ax) or add (Nx) command, the (hexadecimal) null indicator value in the record buffer must be set to

FFFF	the field value is set to "undefined", an insignificant null; the field’s contents in the record buffer is irrelevant when set to binary zero or blank characters.
0000	if either no value is specified in the record buffer, or binary zero or blanks are specified, the field contains a significant null value.

For an *add* command, if no value for the field is supplied in the record buffer for a field defined with the NC option, the field is treated as a null field. The following example shows how a null would be represented in a two-byte Adabas binary field AA defined with the NC option:

Field definition: 01,AA,2,B,NC

	For a nonzero value	**For a blank**	**For null**
Null Value indicator in Record Buffer	0 (binary value is significant)	0 (binary null is significant)	FFFF (binary null is not significant)
Data	0005	0000 (zero)	not relevant
Adabas internal representation	0205	0200	C1

For an *update* (A1/N1) command, the field value is always significant whenever the field is defined with the NC option; the field is treated as if a hexadecimal null indicator value of "0000" has been specified.

For a *read* command, if the null indicator is not specified for an NC option field, the field value is returned in the record buffer whenever there is a significant value in the record. If the Data Storage record contains a "not significant" (FFFF) indicator value for the field, response code 55 will be returned when the record is read.

Specifying a Field with LA (Long Alpha) Option

The LA option is normally used with variable-length data. The following specifications illustrate alpha fields with LA option in the format and record buffers.

The length of an alpha field with LA option can be specified in the record buffer. The field value is preceded by a two-byte length field containing the length of the value, plus 2 (inclusive length).

Format Buffer	`AA, ...`	Name of the fields to be read

Record Buffer	`0005ABC ...`
	or
	`2712 ...(10,000 characters)..`
		Field values returned by Adabas

The length of an alpha field with LA option can also be specified in the format buffer; however, the length is then limited to 253 bytes:

Format Buffer	`AA,5, ...`	Name of the fields to be read
Record Buffer	`ABC__`	Field values returned by Adabas

A field with LA option can also have the NU (null suppression), NC/NN SQL null significance (page), or the MU (multiple-value field) option, and can be a member of a PE (periodic) group.

A field with the LA option cannot be a descriptor and cannot be the parent of sub-/superfields, sub-/superdescriptors, hyperdescriptors, or phonetic descriptors.

A field is compressed the same way, with or without the LA option; that is, by removing trailing blanks. This must be kept in mind if you store binary data in a long alpha field.

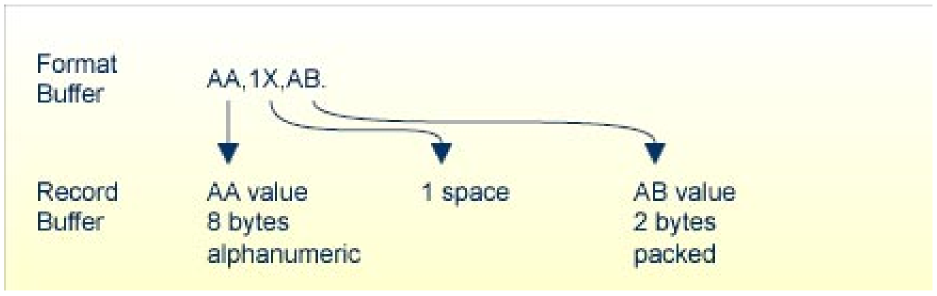
Format and Record Buffer Examples

This section provides examples of format and record buffer construction. For the Adabas file definitions used in all the examples in this section, see *File Definitions Used in Examples*.

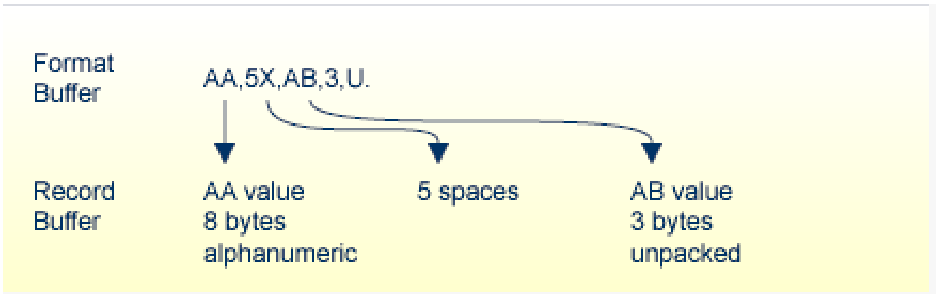
This section covers the following topics:

- Example 1: Using Elementary Fields (Standard Length and Format)
- Example 2: Using Elementary Fields (Length and Format Override)
- Example 3: A Reference to a Periodic Group
- Example 4: The First Two Occurrences of Periodic Group GB
- Example 5: The Sixth Value of the Multiple-Value Field MF
- Example 6: The First Two Values of the Multiple-Value Field MF
- Example 7: The Highest Occurrence Number of a Periodic Group GC and the Existing Number of Values for the Multiple-Value Field MF

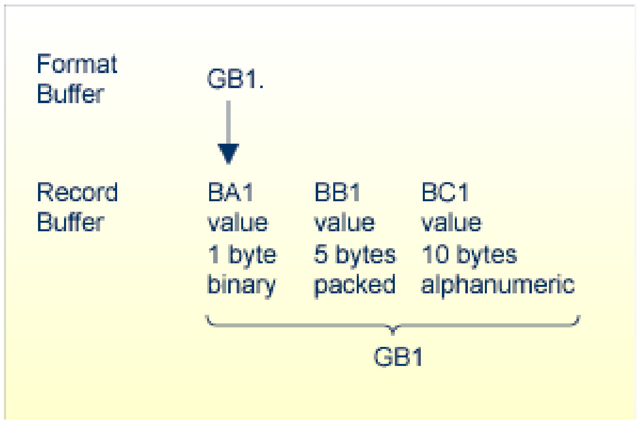
Example 1: Using Elementary Fields (Standard Length and Format)



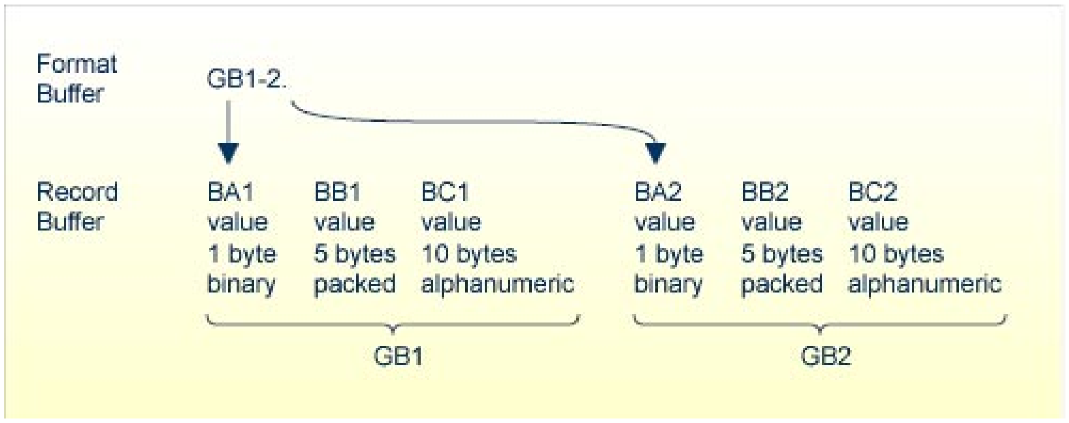
Example 2: Using Elementary Fields (Length and Format Override)



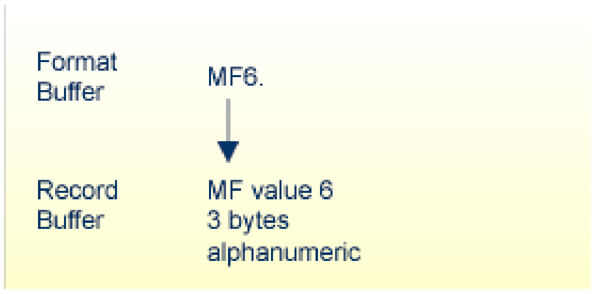
Example 3: A Reference to a Periodic Group



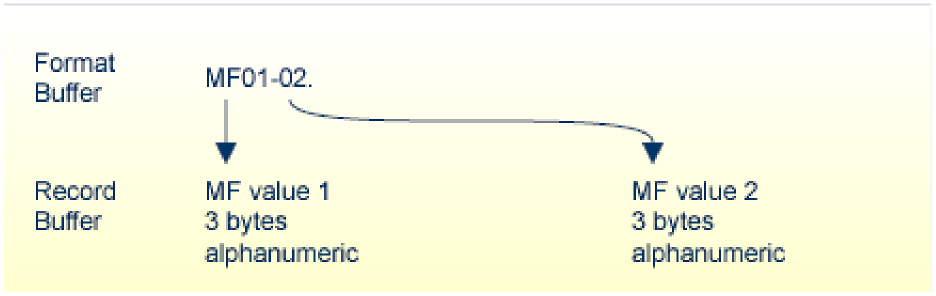
Example 4: The First Two Occurrences of Periodic Group GB



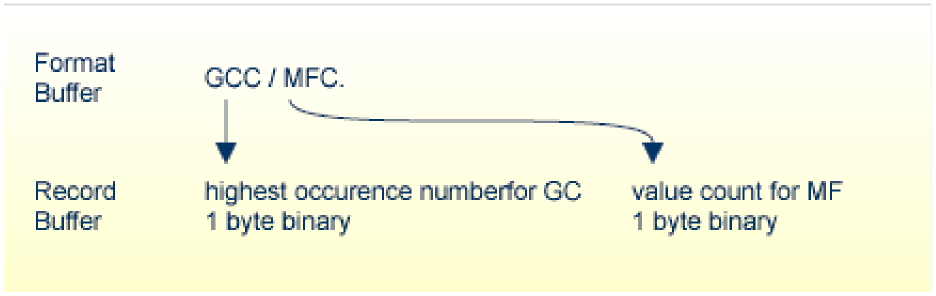
Example 5: The Sixth Value of the Multiple-Value Field MF



Example 6: The First Two Values of the Multiple-Value Field MF



Example 7: The Highest Occurrence Number of a Periodic Group GC and the Existing Number of Values for the Multiple-Value Field MF



Search and Value Buffers

The search and value buffers are used together to define

- the search criterion to select a set of records using a FIND command (S1, S2, S4); and
- the range of values to be traversed by logical sequential read commands (L3/6, L9).

The user provides the search expression(s) in the search buffer and the values which correspond to the search expressions in the value buffer.

The syntax used for the search buffer depends on the type of search criteria to be employed:

1. Single file search. The search criteria consists of one or more fields contained in a single file;
2. Multiple file search using physically coupled files. The search criteria consists of fields contained in two or more files which have been physically coupled using the ADAINV utility;

3. Search using the soft coupling feature. This feature provides for a combination of search, read, and internal list matching.

A search criteria may also contain one or more fields which are not defined as descriptors. If nondescriptors are used, Adabas performs a read operation to determine which records are to be returned to the user.

Note:
On files containing a large number of records, performing a search using nondescriptors can result in excessive response times.

Search Buffer Syntax

Overview

This section outlines the syntax statement options for the search buffer. Delimiters (commas, slashes, parentheses, semicolons) must separate all search buffer entries as indicated. One or more spaces may be present between entries. The search statement must end with a period.

This section covers the following topics:

- Search One File
- Search Multiple, Physically-Coupled Files
- Search One or More Files Using Soft Coupling

Search One File

The following syntax statement is relevant when searching fields in a single file:

```
search-expression [ { , connecting-operator , search-expression } ...] .
```

For the syntax of the "search-expression", see *Search Expression*.

For information about the "connecting-operator", see *Connecting Search Expressions*.

Search Multiple, Physically-Coupled Files

The following syntax statement is relevant for multiple-file searches in which fields from two or more physically coupled files are to be used:

```
/ file-x / search-expression [ {, connecting-operator , search-expression } ... ]
{ , D , / file-y  search-expression /[ { , connecting-operator , search-expression }... ] } ... .
```

For information about search buffer syntax using physically-coupled files, see *Physically Coupled Files*.

Search One or More Files Using Soft Coupling

The following syntax statement is relevant for searching one or more files using soft coupling:

```
( m-file, m-field, s-file, s-field  [ { ; m-file, m-field, s-file, s-field } ... ])
/ s-file-x / search-expression [ { , connecting-operator , search-expression } ... ]
[ { , D ,  / s-file-y /search-expression [ { , connecting-operator , search-expression } ... ] } ... ] .
```

For information about search buffer syntax using soft coupling, see *Soft Coupling*.

Search Expression

The search expression syntax is common to all types of searches:

```
{ field-name [ i > S ] [ , length ] [ , format ] [ , value-operator | EQ }] }
{ ( command-id )                                                            }
```

Search Expression Syntax

The search expression comprises either a field name with optional entries or a command ID.

A command ID value (enclosed within parentheses) identifies a list of ISNs resulting from a previous Sx command that specified the save-ISN-list option.

This section covers the following topics:

- field-name
- Occurrence Index [i]
- S (Significance) and Null Indicators
- length and format
- value-operator

field-name

The search expression can name a field (descriptor or nondescriptor), subdescriptor, superdescriptor, hyperdescriptor, collation descriptor, or phonetic descriptor. When using nondescriptors, multiple-value fields are permitted, but sub-/superfields are not.

If a nondescriptor is used, Adabas reads the entire file in order to determine which records satisfy the search criteria. If only descriptors are used, the inverted lists are used and no reading of records is necessary. Search criteria containing nondescriptors and descriptors may be combined.

If a descriptor field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons and therefore, the record will not be returned in a search. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to initialize the field for each record of the file.

If the descriptor is defined with the NU option (null-value suppression), null values are not stored in the inverted lists; therefore, a search for all the records which have the null value will always result in no records found (even if there are records in Data Storage which contain a null value for the descriptor). This rule also applies to subdescriptors. A superdescriptor value is not stored if any field from which it is derived is defined with the NU option *and* the value of that field is actually null.

Occurrence Index [i]

The occurrence index (i) identifies a particular occurrence of a descriptor or nondescriptor within a periodic group and is used to limit the search to only the values located in the specified occurrence. If no index is provided, the values in all occurrences are searched.

- The index comprises one to three digits; leading zeros are permitted.
- An index is *not* permitted for
  - a superdescriptor derived from a field within a periodic group; or
  - a descriptor that is a multiple-value field, or a sub-/superdescriptor derived from a multiple-value field.

In these cases, the values in all occurrences of the periodic group are searched.

S (Significance) and Null Indicators

For fields with the SQL null value compression option NC, a selection for "null" or "not null" can also be made using an "S" null indicator element similar to that described in the section *The SQL Significance Indicator and Field Series Notation*.

Note:
The NC option cannot be applied to fields with the NU (null-value suppression), FI (fixed storage), MU (multiple-value), or PE (periodic group) options, or to group fields.

The SQL significance ("S") indicator must be added to the field name ("field-nameS") and the corresponding SQL null indicator must be specified in the value buffer.

The following hexadecimal null indicators are allowed as search argument values:

FFFF	select null values
0000	select non-null values

Any other null indicator value causes an Adabas response code 52.

The null indicator (hexadecimal FFFF or 0000) has a standard length of two bytes and fixed-point format; this length and format cannot be overridden.

The "S" indicator can only be used with the equals (=) value-operator; using S with any other value operators causes an Adabas response code 61.

Examples:

The S significance operator is part of the search argument for the field AA.

```
AAS.
```

Select records with the FN field value of packed +1 and the AA field value of null (undefined):

Search Buffer	`FN , 2 , P , D , AAS.`	Search argument
Value Buffer	`001F FFFF`	Field value specification

Note:
Insignificant null values are not stored in the index. This can cause a search-for-null operation to be quite costly for an application program's performance.

Select records with the FN field value of packed +1 and the AA field value of non-null:

Search Buffer	`FN , 2 , P , D , AAS.`	Search argument
Value Buffer	`001F 0000`	Field value specification

length and format

The "length" and "format" of the field/descriptor value as provided in the value buffer may be stated explicitly with these parameters. If the length or format parameter is omitted, the value in the value buffer must comply with the standard length and format of the field/descriptor, as shown in the following *Permitted Data Lengths and Formats* table.

value-operator

A value-operator indicates the logical operation to be performed between the preceding descriptor and its corresponding value in the value buffer.

The following operators may be specified:

EQ (or) =	equals
GE	greater than or equal to
GT (or) >	greater than
LE	less than or equal to
LT (or) <	less than
NE	not equal to

If no value-operator is specified, an "equals" operation is assumed.

Examples:

The following examples show the use of a value-operator:

AA.	AA equals the value specified in the value buffer (the default)
AA,LT.	AA is less than the value specified in the value buffer
AA,GE.	AA is greater than or equal to the value specified in the value buffer.

The NE (not equal to) operator selects all records with the FN field not equal to "MIKE":

Search Buffer	**`FN,4,A,NE.`**	Search argument
Value Buffer	`MIKE`	Field value specification

Replacing the NE operator in the above example with EQ (equal to) would select only FN field values of "MIKE".

Connecting Search Expressions

A "connecting-operator" may be used to connect search expressions. The permissible connecting operators are as follows:

Operator	Description
D	The results of two search expressions are to be combined using a logical AND operation: **`AA,D,AB.`**

O	The results of two search expressions are to be combined using a logical OR operation. The OR operator may only be used to connect search expressions which use the same descriptor:
	
	`AA,O,AA.              valid`
	
	`AA,O,AB.              invalid`

R	Fields or command IDs that point to ISN lists derived from different descriptors are to be combined using a logical OR operation: **`AA,5,A,R,AB,LT.       valid`**

S	A FROM-TO range (inclusive) which involves two search expressions. The same descriptor must be used in both expressions:
	
	`AA,S,AA.              valid`
	
	`AA,S,AB.              invalid`

N	Excludes a single value or a range of values from the immediately preceding FROM-TO range. This operator can only be specified in conjunction with the "S" operator, and must apply to the same field specified in the FROM-TO range. Phonetic descriptors cannot be specified:
	
	`AA,S,AA,N,AA.         valid`
	
	`AA,S,AA,N,AB.         invalid`
	
	`AA,S,AA,N,AA,S,AA.    valid`
	
	`AA,S,AA,N,AA,N,AB.    invalid`

Y	The results of any number of D, O, R, S, and N search operations can be combined using a logical AND operation: **`AA,D,AB,Y,AA,O,AA,Y,AA,S,AA,N,AA,S,AA.`** The Y connecting operator functions like parentheses: only one level is allowed; that is, nested parentheses are not supported. All search expressions connected with the Y operator must apply to the same file.

See the section *Search/Value Buffer Examples* for more examples.

Processing Order for Search Buffer Connecting Operators

If different operators are used within a single search buffer argument, the operators are processed in the following order:

1. Evaluate all "S/N/O" operations, as described in this documentation;
2. Evaluate all "D" operations, if needed;
3. Evaluate all "R" operations, if needed;
4. Evaluate all "Y" operations, if needed.

Example:

`Search Buffer = AA,S,AA,O,AA,D,AB,R,AC,D,AD.`

is equivalent to

`( ( (AA,S,AA),O,AA),D,AB),R,(AC,D,AD)`

`Search Buffer =  AA,D,AB,Y,AA,O,AA,Y,AA,S,AA,N,AA,S,AA.`

is equivalent to

`(AA,D,AB),Y,(AA,O,AA),Y,((AA,S,AA),N,(AA,S,AA))`

Physically Coupled Files

The search buffer for a multiple-file search in which fields from two or more physically coupled files are to be used is

```
/ file-x / search-expression [ {, connecting-operator , search-expression } ... ]
{ , D , / file-y / search-expression [ { , connecting-operator , search-expression } ... ]} ... .
```

The only connecting operator allowed in search expressions for physically coupled files is the AND (D) symbol.

The search expressions can be in any order. The ISN values actually returned are from the coupled file specified by the Adabas control block's file number field; this file is called the "primary" file.

The file number can appear only once for a given file. The file number must immediately precede the search expression (or expressions) referring to that file. A maximum of five (5) files may be specified in a single find command.

All files specified must have been previously coupled using the COUPLE function of the ADAINV utility.

All other syntax elements are entered as described in the sections *Search Expression* and *Connecting Search Expressions*.

Example:

Find the ISNs of all the records in file 1 that contain the three-byte (length override) unpacked decimal (format override) value "+20" in their AB fields, and that are coupled to records in file 2 containing the value "ABCDE" for field RB, which has a standard length of ten bytes and an alphanumeric format.

Search Buffer	`/1/AB,3,U,D,/2/RB.`	Search argument

Value Buffer	character-notation
	`020ABCDEbbbbb`
	hex-notation
	`F0F2F0C1C2C3C4C54040404040`
		Field value specification

Soft Coupling

The search buffer for a search in which soft coupling is to be used is constructed as follows:

```
(m-file, m-field, s-file, s-field  [ { ; m-file, m-field, s-file, s-field } ... ] )
/ s-file-x / search-expression [ { , connecting-operator , search-expression } ... ]
[ { , D ,  / s-file-y / search-expression [ { , connecting-operator , search-expression } ... ] } ... ] .
```

m-file, m-field

"m-file" is the main file. This file must also be specified in the file number field of the Adabas control block. The final resulting ISN list will include ISNs contained in the main file only.

"m-field" is the field in the main file that is to be used as the soft-coupling link field. This field must be a descriptor, subdescriptor, superdescriptor, or hyperdescriptor. It may not be a long alphanumeric field, or be contained within a periodic group.

s-file, s-field

"s-file" is the search file; "s-field" is a field within the search file. For each ISN selected from this search file (according to the search criterion), the field specified as "s-field" will be read. The value of the field will then be used to determine which ISNs in the main file have a matching value.

The field may be a descriptor or nondescriptor; it can be a subdescriptor, superdescriptor, hyperdescriptor, or a long alphanumeric field. It must have the same format as the corresponding "m-field". The standard length may be different. The field may not be contained within a periodic group.

A maximum of 42 soft-coupling criteria may be specified. Search criteria for soft coupling are illustrated in the section *Search/Value Buffer Examples*, examples 15-18.

Value Buffer

In the value buffer, the user specifies the values for each descriptor specified in the search buffer.

If the search expression is a command ID, no corresponding entry is made in the value buffer.

The values provided must be in the same sequence as the corresponding search expressions specified in the search buffer. All values provided must correspond to the standard length and format of the corresponding descriptor unless the user has explicitly overridden the standard length or format in the search buffer.

No intervening blanks or other characters such as a comma can be inserted between values in the value buffer. A period is not required to end the value buffer entry.

SQL Null Values and Indicators

When searching for fields defined with the NC (SQL null "not counted") option, the search buffer field definition must contain a null significance ("S") indicator and the corresponding value buffer argument value must display a two-byte binary null value indicator. See the section *S (Significance) and Null Indicators* for more information and examples of the null value indicator in the value buffer.

Sign Handling

Binary values are treated as unsigned numbers. Fixed-point, unpacked, and packed values are treated as signed numbers. Valid signs which may be provided are as follows:

Fixed

The sign is contained in bit 0 (high-order bit):

```
0 = positive
1 = negative (two’s complement)
```

Example (hex notation with decimal equivalent):

```
00000005 = +5
FFFFFFFB = -5
```

Unpacked

The sign is contained in the four high-order bits of the low-order byte:

```
C or A or F or E = positive (CAFE)
B or D           = negative (BD)
```

Example (hex notation with decimal equivalent):

```
F1F2F3  = +123
F1F2D3  = -123
```

Packed

The sign is contained in the four low-order bits of the low-order byte:

```
A or C or E or F = positive*
B or D           = negative*
```

* If a search value is being provided for a superdescriptor which is derived from a packed field, an F positive sign or a D negative sign must be provided.

Example (hex notation with decimal equivalent):

```
X’123F’ = +123
X’123C’ = +123
X’123D’ = -123
```

Search/Value Buffer Examples

This section contains examples of search and value buffer construction. For the Adabas file definitions used in all the examples in this section, see *File Definitions Used in Examples*. The values for the value buffer are shown in character and/or hexadecimal notation.

Example 1: Using a Single Search Expression

A search that uses a single search expression.

Select the ISNs of all the records in file 1 that contain the value "12345" for field AA, which has a standard length of eight bytes and numeric format.

Search Buffer	`AA.`	Search argument

Value Buffer	character-notation
	`00012345`
	hex-notation
	`F0F0F0F1F2F3F4F5`
		Field value specification

The same search may be performed using "AA,5." (length override) in the search buffer and the value "12345" (without trailing blanks) in the value buffer.

Example 2: Using Search Expressions Connected by AND

A search that uses two search expressions connected by the AND operator.

Select the ISNs of all the records in file 1 that contain the value "12345678" for the field AA, which has a standard length of eight bytes and numeric format, and the value "+2" for the field AB, which has a standard length of two bytes and packed decimal format.

Search Buffer	`AA,D,AB.`	Search argument
Value Buffer	hex-notation `F1F2F3F4F5F6F7F8002C`	Field value specification

Select the ISNs of all the records in file 1 that contain the value "12345678" for the field AA, which has a standard length of eight bytes and numeric format, and the value "+2" for the field AB, which has a length of three bytes (override) and unpacked decimal (override) format.

Search Buffer	`AA,D,AB,3,U.`	Search argument

Value Buffer	character-notation
	`12345678002`
	hex-notation
	`F1F2F3F4F5F6F7F8F0F0F2`
		Field value specification

This second search produces the same result as the first search, but shows the use of the length and format override in the search buffer.

Example 3: Using Search Expressions Connected by OR

A search that uses three search expressions connected by the OR operator.

Select the ISNs of all the records in file 2 that contain any of the values "284", "285", or "290" for the field XB. Length and format overrides are used.

Search Buffer	`XB,3,U,O,XB,3,U,O,XB,3,U.`	Search argument

Value Buffer	character-notation
	`284285290`
	hex-notation
	`F2F8F4F2F8F5F2F9F0`
		Field value specification

Example 4: Using Search Expressions Connected by FROM-TO

A search that uses two search expressions connected by the FROM-TO operator.

Select the ISNs of all the records in file 2 that contain any value in the range "+20" through "+30" for the field XB.

Search Buffer	`XB,S,XB.`	Search argument
Value Buffer	hex-notation `020C030C`	Field value specification

Example 5: Using Search Expression with BUT-NOT

A search that uses three search expressions connected by the FROM-TO and BUT-NOT operators.

Select the ISNs of all the records in file 2 that contain any of the values in the range "+20" through "+30" but not "+27" for the field XB.

Search Buffer	`XB,S,XB,N,XB.`	Search argument
Value Buffer	hex-notation `020C030C027C`	Field value specification

Select the ISNs of all the records in file 2 that contain any of the values in the range "+1" through "+200" or "+500" through "+600" for the field XB.

Search Buffer	`XB,S,XB,O,XB,S,XB.`	Search argument
Value Buffer	hex-notation `001C200C500C600C`	Field value specification

Example 6: Using a Multiple-Value Descriptor

A search in which a multiple-value field is used.

Select the ISNs of all the records in file 1 that contain the value "ABC" for any value of the multiple-value field MF.

Search Buffer	`MF.`	Search argument

Value Buffer	character-notation
	`ABC`
	hex-notation
	`C1C2C3`
		Field value specification

It is not possible to limit the search to a specific occurrence of a multiple-value field. The following search buffer entry is invalid:

Search Buffer	`MF2.`	Search argument

Example 7: Using a Descriptor Within a Periodic Group

A search in which a descriptor within a periodic group is used.

Select the ISNs of all the records in file 1 that contain the value "4" in any occurrence of the descriptor BA (which is contained in a periodic group).

Search Buffer	`BA.`	Search argument
Value Buffer	hex-notation `04`	Field value specification

Select the ISNs of all records in file 1 that contain the value "4" in the third occurrence of the descriptor BA (which is contained within a periodic group).

Search Buffer	`BA3.`	Search argument
Value Buffer	hex-notation `04`	Field value specification

Example 8: Using a Subdescriptor

A search that uses a subdescriptor. SA is a subdescriptor derived from the first four bytes of the field RA.

Select the ISNs of all the records in file 2 that contain the value "ABCD" for the subdescriptor SA.

Search Buffer	`SA.`	Search argument
Value Buffer	hex-notation `C1C2C3C4`	Field value specification

Example 9: Using a Superdescriptor with Alphanumeric Format

A search that uses a superdescriptor with alphanumeric format. SB is a superdescriptor derived from the first eight bytes of the field RA and the first four bytes of the field RB.

Select the ISNs of all the records in file 2 that contain the value "ABCDEFGH1234" for the superdescriptor SB.

Search Buffer	`SB.`	Search argument
Value Buffer	hex-notation `C1C2C3C4C5C6C7C8F1F2F3F4`	Field value specification

Example 10: Using a Superdescriptor with Binary Format

A search that uses a superdescriptor with binary format. SC is a superdescriptor derived from the fields XB and XC.

Select the ISNs of all the records in file 2 that contain the value "+20" for the field XB and the value "123456" for the field XC.

Search Buffer	`SC.`	Search argument
Value Buffer	hex-notation `020FF1F2F3F4F5F6`	Field value specification

Example 11: Using Previously Created ISN Lists

A search that uses previously created ISN lists (identified by their command IDs).

Select the ISNs present in both ISN lists identified by the command IDs "CID1" and "CID2".

Search Buffer	`(CID1),D,(CID2).`	Search argument
Value Buffer	not used	Field value specification

Select the ISNs of all the records in file 1 for which an ISN is present in the ISN list identified by "CID1" and which contain the value "+123" for the field AB.

Search Buffer	(CID1),D,AB,3,U.	Search argument

Value Buffer	character-notation
	`123`
	hex-notation
	`F1F2F3`
		Field value specification

Example 12: Using a Value Operator

A search in which a value operator is used.

Select the ISNs of all the records in file 1 that contain a value greater than "+100" for the field AB.

Search Buffer	**`AB,3,U,GT.`**	Search argument

Value Buffer	character-notation
	`100`
	hex-notation
	`F1F0F0`
		Field value specification

Example 13: Using Both Value and Connecting Operators

A search in which both value and connecting operators are used.

Select the ISNs of all the records in file 1 that contain a value greater than "+100" for the field AB and a value greater than "A" for the field AA.

Search Buffer	**`AB,3,U,GT,D,AA,1,GT.`**	Search argument

Value Buffer	character-notation
	`100A`
	hex-notation
	`F1F0F0C1`
		Field value specification

Example 14: Using Physically Coupled Files

A search using search expressions that refer to physically coupled files.

Select the ISNs of all the records in file 1 that contain the value "+20" for the field AB and are coupled to records in file 2 that contain the value "ABCDE" for field RB. Length and format override are used for field AB.

Search Buffer	`/1/AB,3,U,D,/2/RB.`	Search argument

Value Buffer	character-notation
	`020ABCDEbbbbb`
	hex-notation
	`F0F2F0C1C2C3C4C54040404040`
		Field value specification

Example 15: Using Single Soft Coupling Criterion and Single Search Criterion

The file for which ISNs will be returned is determined by the file number field.

File Number	4	
Search Buffer	`(4,AB,1,AC)/1/AB,S,AB.`	Search argument
Value Buffer	`-------------`	Field value specification

1. Search file 1 for AB = value as provided in value buffer.
2. For each resulting ISN in file 1, read field AC and internally match the value with the corresponding value list for file 4. The resulting ISN list from file 4 is provided in the ISN buffer.

Example 16: Using Single Soft Coupling Criterion and Multiple Search Criteria

The file for which ISNs will be returned is determined by the file number field. The order in which the criteria are specified is arbitrary.

File Number	1	
Search Buffer	`(1,AA,2,AB)/1/AC,D,AE,D,/2/AF,S,AF.`	Search argument
Value Buffer	`-------------`	Field value specification

1. Search file 2 for AF = ... through ... (values in value buffer)
2. For each resulting ISN in file 2, read field AB and internally match the value with the corresponding value list for file 1.
3. Search file 1 for AC = ... and AE = ... (values in value buffer).
4. Match resulting ISN lists from steps 2 and 3. The resulting ISNs are provided in the ISN buffer.

Example 17: Using Multiple Soft Coupling Criteria and Multiple Search Criteria

File Number	1	

Search Buffer	`(1,AA,2,AB; 1,AA,5,BA)`
	
	`/1/AC,D,AE,D,/2/AF,S,AF,D,/5/BC,S,BC`
		Search argument

Value Buffer	- - - - - - - - - - - - -	Field value specification

1. Search file 2 for AF = ... through ... (values in value buffer).
2. For each resulting ISN in file 2, read field AB and internally match the value with the corresponding value list for file 1.
3. Search file 5 for BC = ... through ... (values in value buffer).
4. For each resulting ISN in file 5, read field BA and internally match the value with the corresponding value list for file 1.
5. Search file 1 for AC = ... and AE = ... (values in value buffer).
6. Match resulting ISN lists from steps 2, 4 and 5. The resulting ISNs are provided in the ISN buffer.

Example 18: Using Multiple Soft Coupling Criteria and Multiple Search Criteria with Physical Coupling

Search Buffer	`(1,AA,2,AB)`
	`/1/AC,D,AE,D,/2/AF,S,AF,D,/5/BC,S,BC`
		Search argument

Value Buffer	- - - - - - - - - - - - -	Field value specification

1. Search file 2 for AF = ... through ... (values in value buffer)
2. For each resulting ISN in file 2, read field AB and internally match the value with the corresponding value list for file 1.
3. Search file 5 for BC = ... through ... (values in value buffer)
4. For each resulting ISN in file 5, use the physical coupling lists created by the ADAINV utility to perform ISN matching. This is done since no soft coupling criteria was provided from file 5 to the main file (file 1). If a physical coupling list does not exist, any ISNs from file 5 will not be considered.
5. Search file 1 for AC = ... and AE = ... (values in value buffer).
6. Match resulting ISN lists from steps 2, 4 and 5. The resulting ISNs are provided in the ISN buffer.

ISN Buffer

Adabas returns the ISNs of the records that satisfy the search criteria in the ISN buffer.

The four-byte binary ISNs are provided in ascending sequence. For the S2 or S9 command, the ISNs are provided according to the user-specified sort sequence. If the ISN buffer is not large enough to contain the entire resulting ISN list, Adabas stores the overflow ISNs on the Adabas Work dataset, if requested. These overflow ISNs may then be retrieved at a later time.

If the resulting ISNs are to be read using the GET NEXT option of the L1/L4 command, the ISN buffer is not needed.

The ISN buffer also supplies an ISN list when the ET or BT command specifies the hold ISN (P) option.

If the prefetch feature is invoked with a command option rather than with ADARUN, the ISN buffer also holds prefetched records awaiting processing.

If the multifetch option is used, Adabas returns prefetched records in the record buffer and corresponding record descriptor elements in the ISN buffer. When used with ET/BT commands, the multifetch option releases only the subset of the records held by the current transaction that are specified in the ISN buffer; see the section *Multifetch Operation Processing* for more information.

General Programming Considerations

This section explains several concepts that are important to consider when programming calls.

Internal IDs can be specified to perform important functions during Adabas command execution. In the control block of an Adabas direct call command, the user can specify

- a "command ID", a non-blank, non-zero value that acts as an internal ID for decoded record formats; and optionally
- a "format ID", a separate four-byte internal ID for decoded record formats defined either as user-specific or globally available (a "global format ID") to other users running on the same Adabas nucleus.

The uses of these IDs is explored in detail in this section.

Also described in this section are

- the procedures used to retrieve ISNs from the Adabas Work; and
- the multifetch and prefetch options, which are used to reduce execution time for programs that process large amounts of data in sequential order by reducing the number of system commands needed to the complete the Adabas call.

This chapter covers the following topics:

- Command ID, Format ID, Global Format ID
- ISN List Processing
- Using the Multifetch/Prefetch Feature

Command ID, Format ID, Global Format ID

The command ID, specified in the Adabas control block, performs important functions during Adabas command execution. The command ID is an automatically generated or user-specified nonblank, nonzero value that

- prevents repetitive format buffer decoding by acting as an internal ID for decoded record formats;
- tags ISN lists generated by the Sx command for later access, and saves ISN list overflow.

If desired, a separate internal format ID for decoded record buffer formats can be specified. This value is identified by a flag in the high-order (leftmost) two bits of the first byte of the additions 5 field. Depending on the flag value, the format ID can be user-specific (individual) or available to other users running on the same Adabas nucleus (global).

This section covers the following topics:

- Command IDs for Read Sequential Commands
- Command/Format IDs for Read, Update, and Find Commands
- Using Separate Command ID and Format IDs
- Using a Global Format ID
- Control Block Settings for Command / Format / Global Format IDs
- Command IDs Used with ISN Lists
- Automatic Command ID Generation
- Releasing Command IDs
- Internal Identification of Command IDs
- Examples of Command ID Use

Command IDs for Read Sequential Commands

The read sequential commands (L2/L3, L5/L6, L9) require that a command ID be specified. The command ID is needed by Adabas to return the records to the user in the proper sequence. These command IDs are maintained by Adabas in the table of sequential commands.

The command ID value provided with these commands is also entered and maintained in the internal format buffer pool unless a separate format ID is provided as described in the section *Using Separate Command and Format IDs*. The command ID is released by Adabas when an end-of-file condition is detected during read sequential processing.

Command/Format IDs for Read, Update, and Find Commands

The read commands (L1-L6, L9) and update commands (A1/A4, N1/N2) require a format buffer that specifies the fields to be read or updated. This format buffer must be interpreted and converted into an internal format buffer by Adabas. Using a valid command ID avoids repeated interpretation and conversion by successive commands that use the same format buffer.

A read or update command with a valid command ID causes Adabas to check whether the command ID is in the internal format buffer pool. If the command ID is present, its internal format buffer is used, and no format buffer reinterpretation is required.

Note:
When reading or updating a series of records that use the same format buffer, processing time can be significantly reduced by using a command ID.

Internal format buffers (and the format IDs) resulting from L9 commands can only be used by other L9 commands. Moreover, L9 commands cannot use non-L9 internal format buffers / format IDs.

When reading and updating the same fields (for example, L5 followed by A1), Software AG also recommends that the same command ID be used for both commands (see the A1/A4 and N1/N2 commands for restrictions on using the same format buffer for reading and updating).

If the read-first-record option is used with an S1/S2/S4 command and a command ID is specified, the command ID and the resulting internal format buffer are also stored in the internal format buffer pool.

If the internal format buffer pool is full and a command is received with a command ID without an internal format in the pool, Adabas overwrites the longest unused entry in the pool with the new interpreted format ID. If a command is subsequently received that uses the deleted command ID, the reinterpreted format buffer for that command ID replaces the next-longest unused entry in the pool. For this reason, programs must not change the format buffer between successive read or update commands with the same command ID. Note, however, that use of a command ID does not guarantee that the format buffer is not reinterpreted.

Using Separate Command ID and Format IDs

It is possible to use separate values for command IDs and format IDs. As long as the high-order (leftmost) two bits of the additions 5 field are set to binary ’00’, the command ID is automatically used as the format ID. If, however, the additions 5 field’s high-order two bits are binary ’10’, the fifth through eighth bytes (additions 5 + 4(4)) of the field are used as the format ID. Note that the ID may not start with X’FE’ or X’FF’.

Note:
To identify the format ID as separate from the command ID, non-mainframe Adabas environments expect the first byte of the additions 5 field to be any lowercase letter. When using separate format IDs in a heterogeneous environment, it is important to identify them alike across all platforms used in the system.

Using a Global Format ID

Particularly in an online environment, multiple users of the same program often read or update the same fields of a file and therefore use identical format buffers.

- When you use the *individual* format ID option, Adabas must store the same internal format buffer for each user.

- When you use the *global* format ID option, a single internal format buffer is shared by many users and the need for Adabas to overwrite internal format buffer pool entries is reduced. This option identifies the format buffer to each user by format ID only, rather than by both format ID and terminal ID. A command ID cannot be designated as a global format ID; in addition, the restriction of L9 formats and their IDs being valid only for use by other L9 commands also applies to global formats and IDs.

The global format ID option is activated by setting the high-order (leftmost) two bits of the first byte of the additions 5 field to binary ’11’ (see *Example 3: Using a Global Format ID*). This causes all eight bytes of additions 5 to be recognized as the global format ID.

Note:
To identify the format ID as global in non-mainframe Adabas environments, the first byte of the additions 5 field must be set to be any digit or uppercase letter. When using global format IDs in a heterogeneous environment, it is important to identify them alike across all platforms used in the system.

The first byte of the global format ID may be assigned in the hexadecimal range E2-E9; characters S-Z. All other ranges are reserved for use by Software AG.

The allowable range of values for global format IDs in non-mainframe Adabas environments is hexadecimal 51-5A; characters S-Z.

Warning:
Adabas does not verify the assignment of global format IDs. It is the user’s responsibility to ensure that only global format IDs in the allowable range are assigned. Software AG can neither enlarge the range of global format IDs available to users nor make any changes to its products to resolve a global format ID assignment problem.

A global format ID can be deleted using the RC command and specifying the global format ID in the additions 5 field, as described.

Control Block Settings for Command / Format / Global Format IDs

This section covers the following topics:

- Example 1: Command ID Used as Format ID
- Example 2: Command ID Separate from Format ID
- Example 3: Using a Global Format ID

Example 1: Command ID Used as Format ID

When B’00’ is set in the high-order (leftmost) two bits of the first byte of the additions 5 field, the command ID is automatically used as the format ID.

Example 2: Command ID Separate from Format ID

When B’10’ is set in the high-order (leftmost) two bits of the first byte of the additions 5 field, separate values are used for the command ID and the format ID, and the fifth through eighth bytes of the additions 5 field are used as the format ID.

Example 3: Using a Global Format ID

When B’11’ is set in the high-order (leftmost) two bits of the first byte of the additions 5 field, all eight bytes of the field are used as the global format ID.

Command IDs Used with ISN Lists

If a command ID is specified for any command which results in an ISN list (S1,S2,S4,S5,S8,S9), the command ID value may be used to identify the list at a later time.

- If the save-ISN-list option is used for an Sx command, a command ID must be provided. The save-ISN-list option causes the entire ISN list to be stored on the Adabas Work. ISNs from the list may subsequently be retrieved by an Sx command or by using the GET NEXT option of the L1/L4 command.

- If the save-ISN-list option is not used and an ISN buffer overflow condition occurs (the entire ISN list cannot be inserted in the ISN buffer), the overflow ISNs will be stored on the Adabas Work only if a command ID value was used. In this case, the command ID and the ISN list it identifies will be released by Adabas when all the ISNs have been returned to the user.

Automatic Command ID Generation

Automatic command ID generation may be invoked by specifying a command ID value of X’FFFFFFFF’. This causes the Adabas nucleus to generate command IDs automatically, beginning with X’00000001’ and incrementing by 1 for each new command ID. Automatic command ID generation may not be desirable in all cases; refer to the section *Command/Format IDs for Read, Update, and Find Commands*.

Releasing Command IDs

The user may release a command ID and its associated entries (or ISN list) with a RC command, a CL command, or by using the release-CID option of any Sx command (S1, S2, S4, S5, S8, S9).

The RC command contains options that allow the user to release only those command IDs contained in the internal format buffer pool, the table of sequential commands, or the table of ISN lists.

The CL command causes all the command IDs currently active for the user to be released.

The release-CID option of an Sx command causes the CID specified to be released as the first action taken by the command.

Internal Identification of Command IDs

Each command ID entry is identified by Adabas using an internal user ID together with the command ID value. As a result, one user need not be concerned with the command ID values in use by another. However, a user should avoid using the same command ID value for different commands, particularly if the command ID is used for sequential read (L2/L5, L3/L6, L9) commands and Sx commands.

Examples of Command ID Use

This section covers the following topics:

- Example 1 : Find / Read Processing
- Example 2 : Find / Read Using the GET NEXT Option
- Example 3 : Read / Update Processing
- Example 4 : Read / Find Processing

Example 1 : Find / Read Processing

A set of records is to be selected and read. The same format buffer is to be used for each record being read.

```
FIND (S1)           CID=EX1A
READ (L1)           CID=EX1B
READ (L1)           CID=EX1B
```

Example 2 : Find / Read Using the GET NEXT Option

A set of records is to be selected and read using the GET NEXT option of the L1/L4 command.

```
FIND (S1)           CID=EX2A
READ (L1)           CID=EX2A
READ (L1)           CID=EX2A
READ (L1)           CID=EX2A
```

Example 3 : Read / Update Processing

A file is to be read and updated in sequential order. The same format buffer is to be used for reading and updating.

```
READ PHYS SEQ (L5)  CID=EX3A
UPDATE (A4)         CID=EX3A
READ PHYS SEQ (L5)  CID=EX3A
UPDATE (A4)         CID=EX3A
```

Example 4 : Read / Find Processing

A file is to be read in logical sequence. A find command is to be issued to a second file using the value of a field read from the first file, and the records that result from the find command are then to be read using the GET NEXT option.

```
READ LOG SEQ (L3)   CID=EX4A
FIND (S1)           CID=EX4B
READ (L1)           CID=EX4B
READ (L1)           CID=EX4B
READ LOG SEQ (L3)   CID=EX4A
FIND (S1)           CID=EX4B
READ (L1)           CID=EX4B
```

ISN List Processing

This section discusses the procedures used to retrieve ISNs from the Adabas Work dataset. If the GET NEXT option of the L1/L4 command is used to read the records that correspond to the ISNs contained in the ISN list, ISN handling as discussed in this section is performed automatically by Adabas, and the user need not make use of these procedures.

The notation Sx command as used in this section refers to any command that may result in an ISN list (S1/S2/S4/S5/S8/S9).

This section covers the following topics:

- Storage of ISN Lists
- Retrieval of ISN Lists
- ISN List Processing Examples

Storage of ISN Lists

Adabas stores ISNs on the Work dataset under either of the following conditions:

- An Sx command is issued, a nonblank nonzero command ID is specified, and the save-ISN-list option is specified. The entire resulting ISN list is stored;
- An Sx command is issued, a non-blank non-zero command ID is specified, the save-ISN-list option is not specified, and the resulting ISN list contains more ISNs than can be inserted in the ISN buffer. Only the overflow ISNs are stored in this case.

If an Sx command is issued with blanks or binary zeros in the command ID field, Adabas does not store any ISNs on the Adabas Work.

Retrieval of ISN Lists

The user can retrieve ISNs stored on the Adabas Work dataset by issuing an Sx command in which the same command ID value is used as was used for the initial Sx command. When an Sx command with an active command ID value is issued, Adabas uses this as an indicator that the user is requesting ISNs from an existing ISN list. Adabas locates the ISN list identified by the specified command ID and inserts the next group of ISNs in the ISN buffer. As many ISNs are returned as can fit in the ISN buffer.

This section covers the following topics:

- Save-ISN-List Option Specified
- Save-ISN-List Option Not Specified
- Using the ISN Quantity Field of the Control Block

Save-ISN-List Option Specified

If the save-ISN-list option was specified with the Sx command used to create the ISN list, Adabas uses the ISN specified in the ISN lower limit field to determine the next group of ISNs to be returned.

The next group begins with the first ISN that is greater than the ISN specified in ISN lower limit.

- If binary zeros are specified, the next group begins with the first ISN in the list.
- If a value is specified which is greater than any ISN in the list, response code 25 is returned.

If the ISN list was created using an S2 command, the ISN specified must be present in the ISN list. Use of the save-ISN-list option thus permits the user to skip forward and backward within an ISN list. This is useful for programs that must perform forward and backward screen paging.

Save-ISN-List Option Not Specified

If the save-ISN-list option was not specified with the Sx command used to create the ISN list, Adabas returns the ISNs in the order in which they are positioned in the list, and deletes each group from the Work when it has been inserted in the user’s ISN buffer. The command ID used to identify the list is released when the last group of ISNs has been returned to the user. The ISN lower limit field is not used in this case, unless processing is to begin above a specified ISN range.

Using the ISN Quantity Field of the Control Block

The user can determine when all of the ISNs in a list have been retrieved by using the ISN quantity field of the control block. In this field

- the first Sx command returns the total number of records that satisfy the search criteria.
- each subsequent Sx command used to retrieve ISNs from the Adabas Work returns the number of ISNs that were inserted in the ISN buffer.

ISN List Processing Examples

This section covers the following topics:

- Example 1 : Using Sx Command with L1/L4 Commands with GET NEXT Option
- Example 2 : Using the Save-ISN-List Option
- Example 3 : With ISN Overflow Handling
- Example 4 : Without ISN Overflow Handling

Example 1 : Using Sx Command with L1/L4 Commands with GET NEXT Option

```
Sx command
L1/L4 command with 'GET NEXT' option
L1/L4 command with 'GET NEXT' option
L1/L4 command with 'GET NEXT' option
```

The following examples show various results according to the size of the ISN buffer. Positioning for ISN list is defined by ISN lower limit.

1.
   ```
   ISN Buffer Length = 0
   read the ISN list to the work area
   ```
2.
   ```
   L1/L4 with GET NEXT
   result: first ISN's record
   ```

1.
   ```
   ISN Buffer Length = 4
   read first ISN with S1
   ```
2.
   ```
   L1/L4 with GET NEXT
   result: second ISN's record
   ```

1.
   ```
   ISN Buffer Length = 12
   read first ISN with S1
   first 3 ISNs are returned in ISN buffer
   ```
2.
   ```
   L1/L4 with GET NEXT
   result: read fourth/fifth/sixth ISNs' records
   ```

Example 2 : Using the Save-ISN-List Option

Initial Sx call using save-ISN-list option:

```
Command = Sx
Command ID = SX01               (save-ISN-list option)
Command Option 1 = H
ISN Lower Limit = 0
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7 (total matching ISNs in stored list)

Resulting ISN list: (all ISNs are stored on Work):

```
8    12  14  15  24  31  33
```

Resulting ISN buffer:

```
8    12  14  15  24
```

Subsequent Sx call:

```
Command = Sx
Command ID = SX01
ISN Lower Limit = 24     (limit ISN choice to 24, +)
ISN Buffer Length = 20   (space for 5 ISNs from ISN list)
CALL ADABAS ...
```

Resulting ISN quantity = 2 (total ISNs returned in ISN buffer)

Resulting ISN buffer:

```
31   33   14  15  24.... remainder of ISN buffer unchanged....
```

Subsequent Sx call:

```
Command = Sx
Command ID = SX01
ISN Lower Limit = 0
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7

Resulting ISN buffer:

```
8    12  14  15  24
```

Example 3 : With ISN Overflow Handling

Initial Sx call (save-ISN-list option not used):

```
Command = Sx
Command ID = SX02
Command Option 1 = blank     (no option)
ISN Lower Limit = 0
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7 (total ISNs returned in ISN buffer and stored on Work)

Resulting ISN list: (only ISNs 31 and 33 are stored on Work)

```
8    12  14  15  24  31  33
```

Resulting ISN buffer:

```
8    12  14  15  24
```

Subsequent Sx call:

```
Command = Sx
Command ID = SX02
ISN Lower Limit     (not used)
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 2

Resulting ISN buffer:

```
31   33   14   15  24
```

ISNs 31 and 33 are deleted from the Adabas Work, and command ID SX02 is released. A subsequent Sx call with command ID ’SX02’ will be processed as an initial Sx call since ’SX02’ was released after the last ISNs were returned to the user.

Although the ISN lower limit is not specified in this example, a non-zero value would also return only those ISNs greater than the specified value in the ISN buffer, just as in example 2.

Example 4 : Without ISN Overflow Handling

Initial Sx call with blank or zero command ID:

```
Command = Sx
Command ID = blanks or binary zeros
Command Option 1 = blank     (no option)
ISN Lower Limit = 0          (no lower limit specified)
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7 (total matching ISNs)

Resulting ISN list: none are stored on Work

```
8  12  14  15  24  31  33
```

Resulting ISN buffer:

```
8  12  14  15  24
```

A subsequent Sx call with command ID equal to blanks or binary zeros and ISN lower limit equal to 0 will result in a reexecution of the same find command with the same result as the initial call. A subsequent call with command ID equal to blanks or binary zeros and ISN lower limit = 24 causes reexecution of the Sx command. The result will be ISN quantity of 2 with ISNs 31 and 33 in the ISN buffer.

Using the Multifetch/Prefetch Feature

Programs that process large amounts of data in sequential order require frequent storage access, causing long execution times. The Adabas multifetch and prefetch options significantly reduce the execution times of such programs by reducing the number of system commands needed to complete Adabas calls.

The multifetch and prefetch options reduce execution time in almost all normal applications; however, the specific advantage depends on the type of application program.

Multifetch and prefetch can be invoked with ADARUN parameters or at command level. The ADARUN PREFETCH, PREFTBL, and other related parameters provide control for batch jobs, and the command options M/O (multifetch) and P (prefetch) provide control at the command level. See the *Adabas Operations* documentation for information about the ADARUN PREFETCH multifetch/prefetch control parameter.

This section covers the following topics:

- Multifetching vs. Prefetching
- Invoking Multifetch/Prefetch
- Multifetch Operation Processing
- Prefetch Operation Processing
- Additional Prefetch Programming Considerations

Multifetching vs. Prefetching

The multifetch feature reduces the communication overhead between the application program and the Adabas nucleus. Multifetch buffers multiple record results from a single call, and then transfers the records to the user. Without multifetch, multiple Adabas calls would be necessary to obtain the same result.

Multifetch operation is similar to prefetch; multifetch comprises prefetch functions and more. Adabas 5.3.2 and above support *both* prefetch and multifetch; however, new programs should call the multifetch (M) option, which is common across all Adabas platforms.

Invoking Multifetch/Prefetch

There are two ways to invoke the prefetch option. How it is invoked determines where the preread records are held for processing and therefore what buffer space must be allocated.

- The first method, specifying the PREFETCH=YES or OLD parameter on the ADARUN statement, is the most efficient and requires no application programming changes.

  When PREFETCH=YES or OLD, Adabas uses a double buffering technique that allows processing of one group of records while the following group is being fetched.

  A prefetch buffer must first be defined with the PREFSBL and PREFTBL ADARUN parameters. This buffer must be at least twice the size of the maximum record expected. For more information on specifying these and other ADARUN PREFETCH parameters, refer to the *Adabas Operations* documentation.

- The second method is to specify the M or O option (for multifetch) in the L1/L4, L2/L5, L3/L6, L9, BT or ET commands; or the P option (for prefetch) in the L1/L4, L2/L5, L3/L6 or L9 commands. See the section *Prefetch Operation Processing* for more information.

Multifetch Operation Processing

This section covers the following topics:

- Overview
- READ (Lx) Multifetch Processing
- BT / ET Multifetch Processing

Overview

Multifetch operation is compatible with the corresponding operation on non-mainframe platforms, and can be used across platforms in heterogeneous environments.

Multifetching applies to the following Adabas commands:

- L1/L4 with I or N option (read by ISN, find with GET NEXT)
- L2/L5 (read physical)
- L3/L6 (read logical by descriptor)
- L9 (histogram)
- BT (backout transaction)
- ET (end of transaction)

For all read calls (Lx), multifetch returns a group of records in the record buffer and a description of these records in the caller’s ISN buffer. The maximum number of records is limited by the following values, which are specified in the Adabas control block:

- user-defined maximum as input to the call;
- record buffer length;
- ISN buffer length.

READ (Lx) Multifetch Processing

Prior to the Adabas call, certain fields of the Adabas control block must be set as follows:

Command code	Supported command type and options (see the command list in section *Multifetch Operation Processing*)
ISN lower limit (ISL)	Maximum number of values to return, or "0" to multifetch all values.
Command option 1	Set to "M" or "O" (see note 1 below)
Record buffer length	length of the record buffer
ISN buffer length	length of the ISN buffer

Notes:

1. *Command option "M" indicates that the multifetch option is to be used. Command option "O" selects both the multifetch option ("M") and the existing command option "R" (returns Adabas response code 145 for a requested ISN that is already being held by another user) for the L4/L5/L6 commands.*
2. *For an L1 command, either the command option "I" (ISN sequence) or option "N" (GET NEXT option) must be specified with the multifetch command option "M" or "O"; otherwise, Adabas response code 22 occurs.*

The contents of the returned record buffer and ISN buffer are as follows:

```
Record Buffer: record1,record2, ... ,recordn
```

Records are returned in the record buffer as usual. If more than one record is returned, all records are placed adjoining in the record buffer.

Descriptive elements for these records are returned in the ISN buffer. The first (leftmost) fullword of the ISN buffer contains the number of elements that follow (signed integer, four bytes). Following this count are the record descriptor elements, each 16 bytes long:

```
ISN Buffer: Record descriptor element count{record descriptor element }...
```

A record descriptor element has the structure shown in the following *Multifetch Record Descriptor Element (in ISN Buffer)* table.

Format	Length	Content
All fields unsigned integer, right aligned	4 bytes	Length of this record in record buffer. Records may have different lengths.
	4 bytes	Adabas response for this record. If a nonzero response is given, no record is stored in the record buffer.
	4 bytes	ISN for this record.
	4 bytes	(L9 only) ISN quantity: value count for this descriptor.

If an error is detected while the first record is being processed, the error response is returned in the response code field of the Adabas control block.

If an error is detected while a record other than the first is being processed, the response code is returned in the corresponding record descriptor element in the ISN buffer.

BT / ET Multifetch Processing

By default, Adabas releases all currently held ISNs for the user issuing a BT/ET command. With the multifetch option, only a subset of the records held by the current transaction is released. The records to be released from hold status are specified in the ISN buffer. The first fullword in the ISN buffer specified the number of 8-byte elements following.

You can activate the command-level multifetch feature for the ET/BT command call by setting the following fields of the Adabas control block as indicated:

Command code	BT or ET
Command option 1	M
ISN buffer length	The length of the ISN descriptor element count and all descriptor elements, minimum

Note:
If multifetch is set with ADARUN PREFETCH=YES, the "P" option is automatically used for ET/BT commands (the "M" option is automatically used for all other commands).

The ISN buffer must contain the following values:

```
ISN Buffer: ISN descriptor element count {ISN descriptor element (See table below)} ...
```

where a file number/ISN element has the format shown in the following *Multifetch ISN Descriptor Element for BT/ET Command* table.

Format	Length	Content
Binary, right aligned	4 bytes	Adabas file number
	4 bytes	ISN

Prefetch Operation Processing

Prefetch is effective for programs that use sequential commands (L1/L4 with GET NEXT, L2/L5, L3/L6, L9). When using prefetch, a series of sequential read commands requires only one Adabas call. This single call causes several records to be read at a time from the database. This results in a significant reduction in interregion communication overhead and also permits the overlapped operation of the user program and the Adabas nucleus.

Note:
If the hold option is used (L4/5/6 commands), Adabas places records in hold status when they are read into the prefetch buffer area. This means that if an ET command is issued before all records have been processed, all records (including those not yet processed) are released. The hold ISN option of the ET or HI command can be used to place any such records back into hold status.

Specific commands and/or files can be excluded from prefetch option processing by specifying the files or commands to be excluded with the respective ADARUN PREFXFIL or PREFXCMD parameters.

Invoking Prefetch Operation with Command Option "P"

When enabling prefetch with the command-specific "P" option, Adabas uses the ISN buffer defined within the user program as the intermediate storage area for the pre-read records. Each record in the ISN buffer is preceded by a 16-byte header:

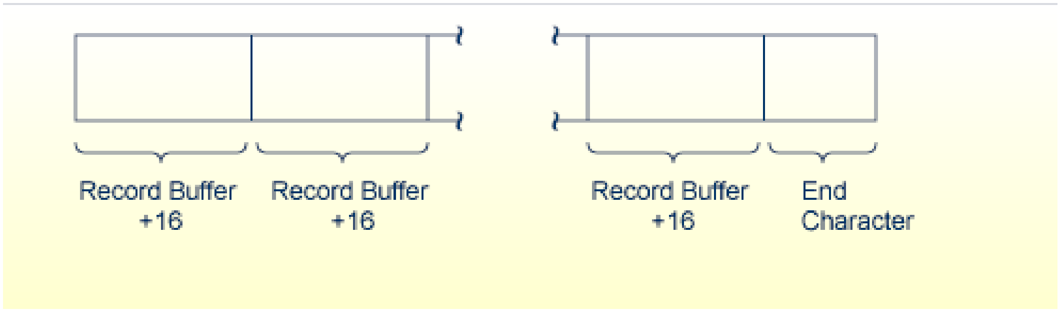
Byte	Use
1-2	Length of record (including length definition). A length of zero indicates the end of data.
3-4	Nucleus response code
5-8	Nucleus internal ID (if the response code is neither zero nor 3, a subcode is returned in the rightmost 2 bytes)
9-12	ISN of the record
13-16	ISN quantity (L9 command only)

The first record is provided by Adabas in the record buffer (without the 16-byte header). The user must then process additional records from the ISN buffer. When end-of-file occurs, the header of the last record in the ISN buffer contains Adabas response code 3, and the two-byte end character contains binary zeros.

Additional Prefetch Programming Considerations

The following are points to consider when using the prefetch option:

- The record buffer size should be set just large enough to contain the largest expected decompressed record.
- If the sequential pass of a file is not to be continued until end-of-file condition is detected, be sure to issue an "RC" command to release the command ID used whenever file processing has been completed.
- The command ID should not be changed during file processing.
- When using a command option "P" to invoke prefetch operation, the ISN buffer size must be a multiple of the total of the record buffer length plus 16, and a final two bytes for an end character:



ISN Buffer Size for Prefetch Programming

Adabas Commands

This part of the Adabas Command Reference documentation contains a detailed description of each Adabas command arranged in alphabetical order by command code (Code).

The information is organized under the following headings:

A1 Command: Update Record	Update record(s) (hold option)
BT Command: Back Out Transaction	Remove database updates for ET logic users
C1 Command: Write a Checkpoint	Write command ID, PLOG, RABN RABN checkpoint, buffer flush option
C3 Command: Write Checkpoint	Write SYNX-03 checkpoint for exclusive control update users; option to store user data
C5 Command: Write User Data to Protection Log	Write user data on SIBA/PLOG
CL Command: Close User Session	End/ET session and update database
E1 Command: Delete Record / Refresh File	Delete record (hold option) or refresh file
ET Command: End Transaction	End and save current transaction
HI Command: Hold Record	Prevent record update by other users
L1 / L4 Command: Read / Read and Hold Record	Read record of specified ISN Read and hold, "wait for held record/issue return code" option
L2 / L5 Command: Read Physical Sequential	Read records in physical order Read in physical order and hold, "wait/issue return code" option
L3 / L6 Command: Read Logical Sequential	Read records in descriptor value order Read in descriptor value order with "wait/issue return code" option
L9 Command: Read Descriptor Values	Read the values of a specified descriptor
LF Command: Read Field Definitions	Read the characteristics of all fields in a file
N1 / N2 Command: Add Record	Add new database record with ISN assigned by Adabas Add new database record with user-assigned ISN
OP Command: Open User Session	Open user session
RC Command: Release Command ID or Global Format ID	Release one or more command IDs or a global format ID for the issuing user
RE Command: Read ETuser Data	Read ET data for this, another, or all users
RI Command: Release Record	Release held record and ISN

● S1 / S2 / S4 Command: Find Records	Return count and ISNs of records satisfying the search criterion Return count of records and ISNs in user-specified order
● S5 Command: Find Coupled ISNs	Return or save a list of coupled ISNs for the specified file
● S8 Command: Process ISN Lists	Combine two ISN lists from the same file with an AND, OR, or NOT operation
● S9 Command: Sort ISN Lists	Sort ISN list in ascending ISN or descriptor-specified sequence

A1 Command: Update Record

The A1 command updates records with a hold option.

This chapter covers the following topics:

- Function and Use
- Command: A1
- Control Block
- Buffers
- Additional Considerations
- Examples

Function and Use

It is used to change the value of one or more fields in a record. The record containing the field (or fields) to be updated is identified by the file number in which it is contained and its ISN. The user specifies the fields to be updated in the format buffer and provides the updating values for these fields in the record buffer. Only the fields specified are modified. All other fields in the record remain unchanged.

All necessary updating to the Associator and Data Storage is performed by Adabas.

A hold option is available to place the record in hold status before it is updated.

If the user is operating in multiuser mode, the A1 command will be executed only if the record to be updated is in hold status for the user.

Note:
The A4 command from previous Adabas releases is executed as an A1 command.

Command: A1

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric / binary	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	U
ISN LOWER LIMIT	17-20	binary	--	A
ISN QUANTITY	21-24	binary	--	A
FORMAT BUFFER LENGTH	25-26	binary	F	U
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-34	--	--	--
COMMAND OPTION 1 / 2	35-36	alphanumeric	F	U
	37-44	--	--	--
ADDITIONS 2	45-48	alphanumeric / binary		A
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
ADDITIONS 5	65-72	alphanumeric	F	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	F	U
RECORD BUFFER	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block

Command Code

A1

Command ID

If a series of records is to be updated by using a series of A1 calls, and the same fields are specified in the format buffer for each call (such as when updating a set of records resulting from a find command), this field should be set to a non-blank, non-zero value. If the A1 command is used in conjunction with an L1/L4, L2/L5, or L3/L6 command, and the same fields within each record are read and updated, the same command ID used for the read command should be used for the A1 calls. In both cases, this reduces the time needed to process each successive A1 call.

If only a single record is to be updated with a single A1 call, or the format buffer is modified between A1 calls, this field should be set to blanks.

The leftmost byte of this field may not be set to hexadecimal ’FF’.

File Number

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

The ISN of the record to be updated.

ISN Quantity/Lower Limit

These fields are set to nulls following completion of the A1 operation.

Format Buffer Length

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified.

Record Buffer Length

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified.

Command Option 1/2 (Hold Record Option)

Option	Description
H	in either of these fields places the record in hold status before it is updated. If the record is currently being held by another user, the command is placed in wait status until either the record becomes available or the transaction times out-unless option "R" is also specified.
R	if specified, must occur in the command option 1 field and option "H" must occur in the command option 2 field. Option "R" causes Adabas to return response code 145 if the record to be held is not available. The command is *not* placed in wait status.

Additions 2 (Length of Compressed Record)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- If the A1 command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Additions 3 (Password)

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

If the accessed file is password protected, Adabas sets this field to blanks during command processing to protect the integrity of the password.

Additions 4 (Cipher Code)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5 (Format ID, Global Format ID)

This field may be used to provide a separate format ID to identify the internal format buffer used for this command, or to provide a global format ID.

As long as the high order bit of the first byte of the additions 5 field is not set to 1, the value provided in the command ID field will be used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the additions 5 field (additions 5 + 4(4)) is used as the format ID.

If the two high-order (leftmost) bits of the first byte of additions 5 field are set to one (B’11’), the value in all eight bytes of the additions 5 field (additions 5 + 0(8)) is used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Buffers

Format Buffer

The fields to be updated must be specified in this buffer. When performing an A1 command, the format buffer cannot contain any of the following:

- A format selection criteria ("fieldname operator value...");
- An edit mask element;
- A reference to a sub-/superdescriptor field;
- The same field specified more than once (except a multiple-value field);
- An "-N" type of specification for an multiple-value field, or for fields in a periodic group (for example, ABN or AB1-N).

Any of the above in the format buffer cause a nucleus response of 44 for an A1 command.

The syntax and examples of format buffer construction are provided in the section *Adabas Calling Procedure*.

Record Buffer

The values to be used for updating are provided in this buffer according to the length and format as specified in the format buffer.

Additional Considerations

The following additional considerations are applicable for the A1 command:

1. Subdescriptors, superdescriptors, and phonetic descriptors may not be directly updated. To update any of the above, the field(s) used to derive the subdescriptor, superdescriptor, or phonetic descriptor must be updated. All corresponding subdescriptor, superdescriptor, or phonetic descriptor values will then be updated by Adabas.

2. Theoretically, the maximum record length permitted is 32767 bytes before compression. The actual maximum is limited by block size restrictions. It is also smaller depending on the size of the LU parameter specified for the Adabas session; the maximum is (LU - format buffer length - 108). The maximum record length after compression is equal to the smaller of either the Data Storage block size - 4 bytes, or the Work block size - 110 bytes.

3. A descriptor value cannot be larger than 253 bytes.

4. If a field is updated using a length override that exceeds the standard length (not permitted if the field is defined with the Fixed Storage option), all subsequent references to this field should specify the length that was used. If a subsequent reference uses the standard length, value truncation of an alphanumeric fields, a response code 55 for a numeric field may occur.

5. A multiple-value field or a field within a periodic group may be specified more than once in the format buffer. A multiple-value field may be specified without an index several times, or with different indices. A periodic field must always have different indices.

6. A multiple-value or periodic group count field specified in the format buffer will be ignored by Adabas. The corresponding value in the record buffer will also be ignored. A literal in the format buffer will be ignored by Adabas, and the corresponding positions in the record buffer will also be ignored.

7. If a multiple-value field is updated and the field count must be changed, Adabas updates the multiple-value field count according to the following rules:

   - For a multiple-value field defined with the null suppression (NU) option, the count field is adjusted to reflect the current number of existing non-null values. Null values are completely suppressed.

Field Definition	01,MF,5,A,MU,NU
MF values before update	XXXXX,YYYYY
Format Buffer	MF4
Record Buffer	ZZZZZ
Result after update	XXXXX,YYYYY,ZZZZZ MF count = 3
MF values before update	XXXXX,YYYYY,ZZZZZ
Format Buffer	MF2
Record Buffer	bbbbb (blanks)
Result after update	XXXXX,ZZZZZ MF count = 2
MF values before update	XXXXX,ZZZZZ
Format Buffer	MF1-2
Record Buffer	bbbbbbbbbb (blanks)
Result after update	Values suppressed MF count = 0

- For a multiple-value field defined without the NU option, the count is adjusted to reflect the current number of existing values (including null values).

Field Definition	01,MF,5,A,MU
MF values before update	XXXXX,YYYYY
Format Buffer	MF4
Record Buffer	DDDDD
Result after update	XXXXX,YYYYY,b(blank),DDDDD MF count = 4
MF values before update	XXXXX,YYYYY,ZZZZZ
Format Buffer	MF3
Record Buffer	bbbbb (blanks)
Result after update	XXXXX,YYYYY,b (blank) MF count = 3

A maximum of 191 values is permitted for a multiple-value field.

An exception to the rules stated on the previous page is when the format buffer contains a multiple-value field *without an index specification*. In this case, only the new values specified are used and all other values are deleted regardless of the content of the value.

Field Definition	01,MF,5,A,MU,NU
MF values before update	XXXXX,YYYYY
Format Buffer	MF
Record Buffer	AAAAA
Result after update	AAAAA MF count = 1
MF values before update	XXXXX,YYYYY
Format Buffer	MF1
Record Buffer	AAAAA
Result after update	AAAAA,YYYYY MF count = 2
MF values before update	XXXXX,YYYYY,ZZZZZ
Format Buffer	MF
Record Buffer	bbbbb
Result after update	value suppressed MF count = 0

8. If one or more fields contained in a periodic group are updated, Adabas updates the periodic group count, if necessary, according to the following rule:

   The count is adjusted to reflect the highest occurrence number referenced in the format buffer (provided that this occurrence is higher than the existing highest occurrence number).

Field Definitions	01,GB,PE 02,BA,1,B,DE,NU 02,BB,5,P,NU
GB values before update	GB (1st occurrence) BA = 5 BB = 20 GB (2nd occurrence) BA = 6 BB = 25 GB count = 2
Format Buffer	GB4
Record Buffer	X’08000000500F’
Result after update	GB (1st occurrence) BA = 5 BB = 20 GB (2nd occurrence) BA = 6 BB = 25 GB (3rd occurrence) BA = 0 BB = 0 GB (4th occurrence) BA = 8 BB = 500 GB count = 4
GB values before update	GB (1st occurrence) BA = 5 BB = 20 GB (2nd occurrence) BA = 6 BB = 25 GB count = 2
Format Buffer	GB1
Record Buffer	X’00000000000F’
Result after update	GB (1st occurrence) BA = 0 BB = 0 GB (2nd occurrence) BA = 6 BB = 25 GB count = 2

A maximum of 191 occurrences is permitted for a periodic group.

9. If a field defined with variable length (no standard length) is specified in the format buffer, the corresponding value in the record buffer must be preceded by a one-byte binary length value that includes the length byte itself.

Field Definitions	01,AA,3,A 01,AB,A
Format Buffer	AA,AB
Record Buffer	X’F1F2F306F1F2F3F4F5’

Fields AA and AB are to be updated. The new value for AA is "123". The new value for AB (which is a variable-length field) is "12345".

Examples

For the Adabas file definitions used in all the examples in this section, see *File Definitions Used in Examples*.

Example 1

ISN 4 of file 1 is to be updated with the following values:

Field AA	1234
Field AB	20

Control Block

Command Code	A1	
Command ID	bbbb	only 1 record is to be updated
File Number	1	
ISN	4	
Format Buffer Length	10	or larger
Record Buffer Length	10	or larger
Additions 3	bbbbbbbb	file is not security protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	AA,AB,2,U.
Record Buffer	X’F1F2F3F440404040F2F0’

Example 2

A set of records (previously identified with a FIND command) in file 2 are to be updated with the following values:

Field RA	ABCD
Field XB	80
Field XC	0

Control Block

Command Code	A1	
Command ID	ABCD	a non-blank, non-zero command ID is recommended because the same fields in a series of records are being updated
File Number	2	
ISN	n	each ISN resulting from the previous FIND command will be inserted in this field before each call
Format Buffer Length	9	or larger
Record Buffer Length	16	or larger
Additions 3	password	file 2 is security protected
Additions 4	bbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	RA,XB,XC.
Record Buffer	X’C1C2C3C440404040080CF0F0F0F0F0F0’

The A1 call is repeated for each ISN which resulted from the previous find command.

BT Command: Back Out Transaction

The BT command removes database updates for ET logic users.

This chapter covers the following topics:

- Function and Use
- Command: BT
- Control Block
- ISN Buffer
- Examples

Function and Use

The BT command is used to remove all database modifications (adds, deletes, updates) performed during the user’s current logical transaction. This may be necessary because of a program error or the determination that the entire transaction cannot be completed successfully. BT commands may only be issued by ET logic users.

Adabas issues an implicit ET command as the last step in the processing of a BT command. This causes the current data protection block to be physically written to the Adabas Work and the data protection log, and the release of all records which were held during the transaction.

The command option 1 field provides the "P" option to place all records listed in the ISN buffer in hold status. The "M" (multifetch) option releases a subset of the records held by the current transaction. The records to be released from hold status are specified in the ISN buffer. The first fullword in the ISN buffer specifies the number of 8-byte elements following.

The command option 2 field provides the "F" (exclude file) option to exclude a single file from backout processing. The updates performed to the file specified will not be backed out. Any records in the file that are in hold status for the user will, however, be released.

Command: BT

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	binary	--	A
FILE NUMBER *	9-10	binary	F *	U
RESPONSE CODE	11-12	binary	--	A
	13-16	--	--	--
ISN LOWER LIMIT	17-20	binary	F	--
	21-32	--	--	--
ISN BUFFER LENGTH **	33-34	binary	F **	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
	37-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
ISN BUFFER **	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Required only if command option 2 is specified
**	Required only if command option 1 is specified
--	Not used

Control Block

Command Code

BT

Command ID

In this field, Adabas returns the transaction sequence number of the transaction that has been backed out. The number is returned in binary format.

File Number

If a file is to be excluded from backout processing, the number of the file to be excluded must be specified in this field, and option F must be specified in the command option 2 field.

If no file is to be excluded (option F is not specified), any value specified in the file number field is disregarded.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

In this field, Adabas returns the response code for the command. Response code 0 indicates that the command was executed successfully. If the BT command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45 - 48 (additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

ISN Lower Limit

If the hold ISNs option is specified, this field must contain the count of six-byte ISN buffer entries.

ISN Buffer Length

The ISN buffer length (in bytes). This length is required only if the hold ISNs or multifetch option is used (see the command option 1 field description). If the multifetch feature is specified, this value must be less than 32 KB.

Command Option 1 (Hold ISNs Option)

Note:
If multifetch is set with ADARUN PREFETCH=YES, the "P" option is automatically used for ET/BT commands (the "M" option is automatically used for all other commands). You *cannot* override this option setting by using this field.

By default as part of BT command execution, Adabas releases all ISNs currently held by the user.

Option	Description
P	places all or a portion of these ISNs back into hold status. The ISNs to be returned to hold status must be provided in the ISN buffer, and the ISN count must be specified in the ISN lower limit field.
M	(command-level multifetch) releases only a subset instead of all of the ISNs held by the current transaction. The records to be released from hold status are specified in the ISN buffer. The first fullword in the ISN buffer specifies the number of 8-byte elements following.

Command Option 2 (Exclude File Option)

Option	Description
F	(exclude file) excludes the single file specified in the file number field from backout processing. The updates performed to the file specified are not backed out. However, any records in the file that are in hold status for the user are released.
blanks	all files are to be included in backout processing.

ISN Buffer

If the command option 1 field is set to "P", each ISN whose record is to be returned to hold status must be provided as a six-byte binary entry in which

- the first two bytes specify the number of the file containing the record; and
- the next four bytes contain the ISN of the record to be held.

If the command option 1 field is set to "M", only a subset of the records held by the current transaction are to be released. The first fullword in the ISN buffer specifies the number of 8-byte elements, and each following eight-byte group is interpreted as one file number/ISN identifier of records to be released from hold status (see the section *BT/ET Multifetch Processing*).

Examples

Example 1

The current user transaction is to be backed out. All files are to be included in the backout process.

Control Block

Command Code	BT	
Command Option 1	blank	no ISNs are held
Command Option 2	b	file exclude option not used

Example 2

The current user transaction is to be backed out. Updates made to file 4 are not to be included in the backout process.

Control Block

Command Code	BT	
File Number	4	file 4 to be excluded from backout
Command Option 2	F	file exclude option used

Example 3

The current user transaction is to be backed out. ISNs 1, 2, and 3 which are contained in file 6 are to be placed into hold status.

Control Block

Command Code	BT	
Command Option 1	P	place ISNs into hold status option
Command Option 2	b	file exclude option not used

Buffer Areas

ISN Buffer	
000600000001	ISN 1
000600000002	ISN 2
000600000003	ISN 2

C1 Command: Write a Checkpoint

The C1 command writes the command ID, PLOG, RABN checkpoint, and buffer flush option.

This chapter covers the following topics:

- Function and Use
- Command: C1
- Control Block
- Example

Function and Use

The C1 command is used to request that a checkpoint be taken.

C1 commands are normally only issued by exclusive control update users (who are not using ET logic) and/or users who are operating in single-user mode.

Adabas executes an implied C1 command at the beginning of a user program in which exclusive file control updating has been requested.

The result of the C1 command is a checkpoint entry in the Adabas checkpoint table. This checkpoint entry

- contains the checkpoint identifier (the value provided by the user in the command ID field), and the current data protection log and block number.
- may be used to restore the database (or certain files) to the status in effect at the time the checkpoint was taken. This may be necessary before a program performing exclusive control updating can be rerun or restarted.

Specifying the "F" (flush buffers) option in either command option field forces Adabas to flush the contents of the Adabas buffer pool to external storage at the end of command processing.

Command: C1

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	A
	9-10	--	--	--
RESPONSE CODE	11-12	binary	--	A
	13-34	--	--	--
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
	37-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Not used

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block

Command Code

C1

Command ID

A non-blank, non-zero value must be entered in this field. This value identifies the checkpoint taken. It is not necessary that each value provided for each checkpoint be unique.

A value of "0000" or "SYNC" may not be specified.

The first byte of this field may not be set to hexadecimal ’FF’.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. If the C1 command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45-48 (additions 2 field), may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Command Option 1/2 (Flush Buffers Option)

Option	Description
F	(flush buffers) specified in either command option field forces Adabas to flush the contents of the Adabas buffer pool to external storage at the end of command processing.

Example

A checkpoint entry is to be written. The checkpoint is to be identified by the value "UCP4".

Control Block

Command Code	C1	
Command ID	UCP4	checkpoint identifier

C3 Command: Write Checkpoint

The C3 command writes a SYNX-03 checkpoint in the Adabas checkpoint file.

This chapter covers the following topics:

- Function and Use
- Command: C3
- Control Block
- Record Buffer
- Example

Function and Use

The C3 command may be issued only by exclusive control/update users (who are not using ET logic).

The primary function of the C3 command is to write a SYNX-03 checkpoint in the Adabas checkpoint file. This checkpoint entry

- contains the current data protection log and block number.
- may be used to restore the database (or certain files) to the status in effect at the time the checkpoint was taken. This may be necessary before a program performing exclusive control updating can be rerun or restarted.

If command option 2 is specified, the C3 command also stores user data in the Adabas checkpoint file for restart purposes. The stored data may be subsequently read with an OP or RE command.

Command: C3

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
	5-10	--	--	--
RESPONSE CODE	11-12	binary	--	A
	13-26	--	--	--
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-35	--	--	--
COMMAND OPTION 2	36	alphanumeric	F	U
	37-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	*	
RECORD BUFFER	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block

Command Code

C3

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully.

Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Record Buffer Length

This field is used only if command option 2 is specified.

The number of bytes of user data to be stored must be specified in this field.

The maximum length that may be specified is 2000 bytes.

Command Option 2: Store User Data

Option	Description
E	indicates that user data is to be stored in the Adabas checkpoint file when the session terminates. This option is only available if the user provided a non-blank, unique user ID during the OP command for the user session.

Record Buffer

The record buffer contains the user data to be stored in the Adabas checkpoint file. The number of bytes actually stored is determined by the value specified in the record buffer length field.

The user data is retained only if the user issued an OP command for the user session in which a non-blank, unique user ID was provided. The data is retained until the next C3 or CL command issued by this user in which user data is provided. If a non-blank, unique user ID was not provided, the data cannot be retrieved in a subsequent session.

Example

The user program issues a C3 command and provides user data to be stored in the Adabas checkpoint file.

Control Block

Command Code	C3	
Record Buffer Length	17	17 bytes of user data to be stored
Command Option 2	E	user data are to be stored

Buffer

Record Buffer	EXU-USER ET-DATA

C5 Command: Write User Data to Protection Log

The C5 command writes user data on SIBA/PLOG.

This chapter covers the following topics:

- Function and Use
- Command: C5
- Control Block
- Record Buffer
- Example

Function and Use

The C5 command is used to write user data to the Adabas data protection log. These data may be read and displayed by using the ADASEL utility.

The data that are written have no effect on Adabas recovery processing. The ADASAV and ADARES utilities ignore all data written to the data protection log as a result of a C5 command.

Command: C5

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
	5-10	--	--	--
RESPONSE CODE	11-12	binary	--	A
	13-26	--	--	--
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	*	
RECORD BUFFER	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block

Command Code

C5

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. If the C5 command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45-48 (additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Record Buffer Length

The number of bytes specified in this field will be written to the Adabas data protection log.

The maximum length which may be specified is 2000 bytes.

Record Buffer

The information to be written to the data protection log is provided in this buffer.

The information written may be selected subsequently using the ADASEL utility by specifying the beginning characters (1 to 30 characters) as originally contained in the record buffer. It is, therefore, recommended that the user provide a unique identification of his data in the beginning positions of the record buffer. This will ensure that the user data can be properly identified and selected.

Example

The information "ULRR0422 UPDATES FOR JANUARY" is to be written to the Adabas data protection log.

Control Block

Command Code	C5
Record Buffer Length	28

Buffer Areas

Record Buffer	ULRR0422 UPDATES FOR JANUARY

CL Command: Close User Session

The CL command ends an ET session and updates the database.

This chapter covers the following topics:

- Function and Use
- Command: CL
- Control Block
- Record Buffer
- Examples

Function and Use

The CL command is used to terminate a user session. Software AG recommends that all user programs issue a CL command upon completion of database processing. User programs operating in single-user mode which are performing database updating must issue a CL command to ensure that all updates have been written to the database.

A CL command

- issues an implicit ET command (ET-logic users only);
- stores user data in an Adabas system file (optional);
- releases all records currently in hold status for the user, and all command ID entries (and corresponding ISN lists) assigned to the user;
- transfers the user’s ET data from the Adabas Work to an Adabas system file. This is done only if a user ID was provided with the OP command; otherwise, any ET data stored during the session is lost.

A CL command issued by a user operating in single-user mode causes a physical close of the database (Associator, Data Storage, Work, and the data protection log). It is therefore not possible for a single-user mode user to follow a CL command with another command (for example, an OP command).

Command: CL

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	binary	--	A
	9-10	--	--	--
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN LOWER LIMIT	17-20	binary	--	A
ISN QUANTITY	21-24	binary	--	A
	25-26	--	--	--
RECORD BUFFER LENGTH	27-28	binary	F *	U
	29-35	--	--	--
COMMAND OPTION 2	36	alphanumeric	F	U
	37-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER *	**	--
RECORD BUFFER *	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Required only if user data to be stored
**	Not used but must be included in parameter list of call statement if user data to be stored
--	Not used

Control Block

Command Code

CL

Command ID

For ET logic users, Adabas returns the transaction sequence number of the user’s last successfully executed transaction in this field. The number is provided in binary format.

Because the CL command includes an ET command (see *Function and Use*), it also increments the transaction sequence number by one.

Response Code

In this field, Adabas returns the response code for the command. Response Code 0 indicates that the command was executed successfully. If the CL command returns a nonzero response code, the rightmost two bytes of the Adabas control block, additions 2 field (bytes 47 and 48) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

ISN: Number of I/Os

Adabas returns the number of I/O operations resulting from this session’s Adabas calls in this field.

ISN Lower Limit: Number of Commands

In this field, Adabas returns the number of commands issued by the user during the user session.

ISN Quantity: CPU Time

In this field, Adabas returns the amount of processor time used by this user for Adabas command processing.

The time is provided in units of 1.048576 seconds.

Record Buffer Length

If user data are to be stored in an Adabas system file, the length of the record buffer must be specified in this field. The length specified determines the number of bytes of user data to be stored.

The maximum length which may be specified is 2000 bytes.

If no user data are to be stored, this field is not used.

Command Option 2: Store User Data

Option	Description
E	indicates that user data is to be stored in an Adabas system file.

Record Buffer

The user data to be stored are provided in this buffer. The number of bytes actually stored is determined by the value specified in the record buffer length field. The data is retained only if the user has issued an OP command in which a nonblank, unique user ID was provided. If so, the data is retained until the next ET or CL command is issued by this user in which user data are provided. If a nonblank user ID was not provided, the data cannot be retrieved in a subsequent session.

Examples

Example 1

The user program has completed all database activity and issues the CL command. No user data are to be stored.

Control Block

Command Code	CL	
Command Option 2	b	no user data are to be stored

Example 2

The user program issues a CL command and provides user data to be stored in an Adabas system file.

Control Block

Command Code	CL	
Record Buffer Length	17	17 bytes of user data to be stored
Command Option 2	E	user data are to be stored

Buffer

Record Buffer	USER 7 NORMAL END

E1 Command: Delete Record / Refresh File

The E1 command deletes a record with the hold option, or refreshes a file.

This chapter covers the following topics:

- Function and Use
- Command: E1
- Control Block
- Examples

Function and Use

The E1 command deletes a record when the record’s ISN is specified, or refreshes a file when an ISN value of zero is specified.

You must specify the file number and ISN of the record to be deleted. Adabas deletes the record from Data Storage. If no ISN is specified and the command ID field contains no command ID (that is, is set to spaces) and the specified file was last loaded with the ADALOD parameter PGMREFRESH=YES, the E1 command refreshes the specified file by first deleting all records of the file and reducing the Associator and Data Storage components of the file to a single extent.

Whether deleting a record or refreshing the entire file, the E1 command also makes the necessary changes to the Associator. You cannot perform both a delete record and file refresh in the same E1 command operation.

If the user is operating in multiuser mode and the record to be deleted is not in hold status for the user, Adabas will place the record in hold status for the user. If the record is in hold status for another user, the E1 command is placed in wait status until the record becomes available. If the "R" option is specified in the command option 1 field and the requested record is not available, response code 145 is returned.

Note:
The E4 command supported in earlier Adabas releases is executed as an E1 command.

Command: E1

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric / binary	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	U
ISN LOWER LIMIT	17-20	binary	--	A
ISN QUANTITY	21-24	binary	--	A
	25-34	--	--	--
COMMAND OPTION 1	35	alphanumeric	F	U
	36-48	--	--	--
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
	65-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Not used

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block

Command Code

E1

Command ID

To refresh a file, set this field to blanks and the ISN field to zero. If you set this field to any other value while specifying an ISN field of zero, the E1 command attempts to remove the record for ISN 0 from the specified file, which causes response code 114.

File Number

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. If the E1 command returns a nonzero response code, the rightmost two bytes of the Adabas control block additions 2 field, bytes 47 and 48, may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

ISN

The ISN of the record to be deleted. If the command ID field contains blanks and this field contains zero, the file specified by the file number field is refreshed. If the command ID field contains non-blanks and this field contains zero, a response code 114 occurs.

ISN Quantity/Lower Limit

These fields are set to nulls following completion of the E1 command operation.

Command Option 1: Response Code 145 if Record not Available

If the user is an ET logic user and the record to be deleted is not in hold status for the user, Adabas places the record in hold status for the user.

If the record to be deleted is being held by another user, the action taken by Adabas is controlled by the setting of the command option 1 field:

Option	Description
R	(return) causes Adabas to return response code 145 if the record to be deleted is not available. The command is not placed in wait status.

Otherwise, Adabas places the E1 command in wait status until either the record becomes available or the transaction times out.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Examples

Example 1

ISN 4 in file 2 is to be deleted.

Control Block

Command Code	E1	
File Number	2	record to be deleted is in file 2
ISN	4	record with ISN 4 to be deleted
Additions 3	password	file 2 is security-protected

Example 2

The file specified in the file field (4) is to be refreshed

Control Block

Command Code	E1	
File Number	4	refresh file 4
ISN		set to binary zero
Command ID	bbbb	set to blanks

The E1 command refreshes file 4.

ET Command: End Transaction

The ET command ends and saves the current transaction.

This chapter covers the following topics:

- Function and Use
- Command: ET
- Control Block
- Buffers
- Examples

Function and Use

The ET command is used to indicate the end of a logical transaction. Each logical transaction should be secured by issuing an ET command. Subsequent restoring or backing out of any later transaction returns the database status to that set by the last successful ET command.

The ET command

- writes all current data protection information to the Adabas data protection log and Adabas Work for all update commands executed successfully during the transaction. If the current session ends abnormally, Adabas uses this protection information to reapply the updates for this transaction to the Associator and Data Storage during the next session;
- releases all records held by the user during the current transaction. ISN lists which were saved by the user as a result of find commands are not released (an option is also available by which these ISNs can be returned to hold status);
- optionally stores user data in an Adabas system file. This data may be read subsequently with an OP or RE command as part of a program restart procedure;
- returns a unique sequence number for the transaction in the command ID field of this and any following OP or CL commands issued by the same user. This sequence number may be used to identify the last successfully processed transaction for the user.

The successful execution of an ET command guarantees that all the updates performed during the transaction will be applied to the database, regardless of any subsequent user or Adabas session interruption.

If an ET logic user issues an OP command after a system failure or in the middle of a transaction, Adabas automatically issues a BT command.

Command: ET

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	binary	--	A
	9-10	--	--	--
RESPONSE CODE	11-12	binary	--	A
	13-16	--	--	--
ISN LOWER LIMIT	17-20	binary	F	--
ISN QUANTITY	21-24	binary	--	A
	25-26	--	--	
RECORD BUFFER LENGTH	27-28	binary	F *	U
	29-32	--	--	--
ISN BUFFER LENGTH **	33-34	binary	F **	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
	37-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER *	***	--
RECORD BUFFER *	F	U
ISN BUFFER **	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Required only of ET data to be stored
**	Required for hold ISN option; optional for multifetch option
***	Not used but must be included in parameter list of call statement if ET data is to be stored
--	Not used

Control Block

Command Code

ET

Command ID

Adabas returns either binary zeros or the transaction sequence number in this field.

If the ET command completes a transaction without update commands (that is, A1, E1, N1, N2), Adabas returns binary zeros.

Otherwise, Adabas returns the transaction sequence number in binary format. These numbers are assigned in ascending sequence during a given user session, starting with 1.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. If the ET command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45-48 (additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

ISN Lower Limit

If the hold ISNs option is specified, this field must contain the count of six-byte ISN buffer entries.

ISN Quantity

In this field, Adabas returns the transaction duration time specified in timer units (each unit equals 1.05 seconds).

Record Buffer Length

If user data are to be stored in an Adabas system file, the number of bytes of user data to be stored must be specified in this field.

Adabas will store the number of bytes specified in this field. The maximum number of bytes which may be specified is 2000 bytes.

If no user data are to be stored, this field is not used.

ISN Buffer Length

The ISN buffer length (in bytes). This length is required only if the hold ISNs or multifetch option is used (see the command option 1 field description).

Command Option 1: Hold ISNs Option

Note:
If multifetch is set with ADARUN PREFETCH=YES, the "P" option is automatically used for ET/BT commands (the "M" option is automatically used for all other commands). You *cannot* override this option setting by using this field.

By default as part of ET command execution, Adabas releases all ISNs currently held by the user.

Option	Description
P	places all or a portion of the ISNs back into hold status. The ISNs to be returned to hold status must be provided in the ISN buffer, and the ISN count must be specified in the ISN lower limit field.
M	(command-level multifetch) releases only a subset of the records held by the current transaction. The records to be released from hold status are specified in the ISN buffer. The first fullword in the ISN buffer specifies the number of 8-byte elements and each following eight-byte group is interpreted as one file number/ISN identifier of records to be released from hold status.

Command Option 2: Store User Data

Option	Description
E	indicates that user data is to be stored in an Adabas system file.

Buffers

Record Buffer

The user data to be stored in an Adabas system file are provided in this buffer.

The data will be retained until the next ET or CL command containing ET data is issued by this user. The user data will be retained when the user session terminates only if the user issued an OP command in which a unique, non-blank user ID was provided.

ISN Buffer

If the command option 1 field is set to "P", each ISN to be returned to hold status must be provided as a six-byte binary entry in which the first two bytes contain the number of the file in which the record is contained, and the next four bytes contain the ISN of the record to be held. Neither the file numbers nor the ISNs are checked for validity.

If the command option 1 field is set to "M", only a subset of the records held by the current transaction are to be released. The first fullword in the ISN buffer specifies the number of 8-byte elements, and each following eight-byte group is interpreted as one file number/ISN identifier of records to be released from hold status (see on page).

Examples

Example 1: ET without User Data

Control Block

Command Code	ET	
Command Option 2	b	no user data are to be stored

Example 2: ET with User Data

Control Block

Command Code	ET	
Record Buffer Length	25	25 bytes of user data to be stored
Command Option 2	E	user data to be stored

Buffer Areas

Record Buffer	USER DATA FOR TRANSACTION

Example 3: ET with Hold ISN Option

Control Block

Command Code	ET	
ISN Lower Limit	3	
ISN Buffer Length	18	
Command Option 1	P	place ISNs in hold status

Buffer Areas

ISN Buffer	
000600000001	ISN 1
000600000002	ISN 2
000600000003	ISN 2

HI Command: Hold Record

The HI command prevents record update by other users.

This chapter covers the following topics:

- Function and Use
- Command: HI
- Control Block
- Example

Function and Use

The HI command is used to place a record in hold status. This command is used to hold a record for subsequent updating without allowing other users to update the record until it is released.

The user specifies the file number and ISN of the record to be held.

If the record to be held is currently being held by another user, the action taken by Adabas is controlled by the setting of the command option 1 field in the Adabas control block. If the command option 1 field

- contains an "R", Adabas returns response code 145 if the record to be held is not available;
- does not contain an "R", Adabas places the user in wait status until the record becomes available, at which time the user is reactivated automatically.

Command: HI

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
	5-8	--	--	--
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	U
	17-34	--	--	--
COMMAND OPTION 1	35	alphanumeric	F	U
	36-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Not used

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block

Command Code

HI

File Number

The number of the file that contains the record to be held.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

In this field, Adabas returns the response code for the command. Response code 0 indicates that the command was executed successfully. If the HI command returns a non-zero response code, the rightmost two bytes of Adabas control block bytes 45-48 (additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

ISN

The ISN of the record to be placed in hold status.

Command Option 1: Response Code 145 if Record Not Available

If the record to be held is currently being held by another user, the action taken by Adabas is controlled by the setting of the command option 1 field:

Option	Description
R	(return) causes Adabas to return response code 145 if the record to be held is not available. The command is not placed in wait status.

Otherwise, Adabas places the command in wait status until either the record becomes available, at which time command and user are reactivated automatically, or the transaction times out.

Example

The record identified by ISN 3 in file 2 is to be placed in hold status. Control is not to be returned until the record is available.

Control Block

Command Code	HI	
File Number	2	record to be held is in file 2
ISN	3	record with ISN 3 to be held
Command Option 1	b	response code 145 option not used

L1 / L4 Command: Read / Read and Hold Record

The L1 and L4 commands read a single record from Data Storage.

This chapter covers the following topics:

- Function and Use
- Command: L1 / L4
- Control Block
- Buffers
- Examples

Function and Use

The user specifies the file number and ISN of the record to be read. The user indicates in the format buffer which fields are to be read. Adabas returns the requested field values in the record buffer.

The L4 command performs the same function as the L1 command, and, in addition, places the record in hold status. If the record to be held is currently being held by another user, the command is placed in wait status until either the record becomes available or the transaction times out. If the L4 command is issued with the command option 1 field set to "R", Adabas returns response code 145 if the record to be read and held is unavailable.

The first unused ISN (F) option specified in the command option 2 field returns the next highest unused ISN for the specified file in the ISN field. The "F" option cannot be used for expanded files.

The multifetch/prefetch option can improve performance by eliminating the time needed for single-record fetches. Multifetching/prefetching can be enabled by specifying "M", "O" (for multifetching) or "P" (for prefetching) in the command option 1 field. Refer to the section *Using the Multifetch/Prefetch Feature* on page for more information.

The GET NEXT (N) option specified in the command option 2 field reads the records identified by the ISNs contained in an ISN list (which was created previously by an Sx command) without the user having to provide the ISN of the record to be read with each L1/L4 call. Adabas selects the ISN from the list and reads the record identified by that ISN.

The read by ISN sequence (I) option specified in the command option 2 field reads a record identified by the ISN you specify in the ISN field. If the ISN specified is not present in the file, the record of the next higher ISN is read, and that record’s ISN is returned in the ISN field.

The compressed option (set by specifying "C." in the format buffer) can be used to request that the record read is to be returned in compressed format as it is stored internally by Adabas.

Command: L1 / L4

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	U *
ISN LOWER LIMIT	17-20	binary	F	U
	21-24	--	--	--
FORMAT BUFFER LENGTH	25-26	binary	F	U
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-32	--	--	--
ISN BUFFER LENGTH **	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
	37-44	--	--	--
ADDITIONS 2	45-48	binary / binary	--	A
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
ADDITIONS 5	65-72	alphanumeric	F	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	F	U
RECORD BUFFER	--	A
ISN BUFFER **	--	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Except for special options
**	The ISN buffer and length required only if the multi-/prefetch option is specified
--	Not used

Control Block

Command Code

L1/L4

Command ID

If a series of records is to be read with a series of L1/L4 calls, and the same fields are to be specified in the format buffer for each call, this field should be set to a non-blank, non-zero value. This results in a reduction of the time required to process each L1/L4 call.

If the GET NEXT option is to be used, the command ID of the ISN list to be used must be specified in this field. The format buffer may not be changed between successive L1/L4 calls when the GET NEXT option is used.

If only a single record is to be read, or if the format buffer is to be modified between L1/L4 calls, this field should be set to blanks.

If the command ID value is X’FFFFFFFF’, automatic command ID generation will be in effect. In this case, the Adabas nucleus will generate values for command ID beginning with X’00000001’, and will increment the value by 1 for each L1/L4 call. When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the additions 5 field for separate format ID and/or global format ID usage.

File Number

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Nonzero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

Response code 3 indicates an end-of-list condition (applicable only if the GET NEXT option is used).

ISN

Specify the ISN of the record to be read.

If you specify the GET NEXT (N) option, Adabas selects ISNs from the ISN list identified by the command ID, reads the record of that ISN, and returns the ISN for the next record in this field; any specified ISN value is ignored.

If you specify the ISN sequence (I) option, you must also specify an ISN value in this field. The L1/L4 command returns that ISN’s data record in the record buffer.

If you specify the "F" option, the L1/L4 command returns the next higher available ISN recorded in the file control block (FCB) in this field.

When a record is read, Adabas returns that record’s ISN in this field, regardless of the option selected. With the option "F", this field returns the next higher unused ISN.

ISN Lower Limit: Multifetch Record Count

If either "M" or "O" (multifetch option) is specified in the command option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and ISN buffer lengths. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

Format Buffer Length

The format buffer length (in bytes). The actual format buffer area defined in the user program must be at least as large as the length specified.

Record Buffer Length

The record buffer length (in bytes). The actual record buffer area defined in the user program must be at least as large as the length specified.

ISN Buffer Length: With Command-Level Multifetch/Prefetch Option Only

The ISN buffer length (in bytes). The ISN buffer defined in the user program must be at least as large as the length specified.

Command Option 1

Option	Description
F	(first unused ISN) returns the next highest unused ISN for the specified file in the ISN field. The "next unused ISN" is determined by referring to the file control block (FCB). Do not use the "F" option when reading Adabas expanded files.
M	(multifetching)
O	(multifetching with the "R" option)
P	(prefetching)
	Specifying one of these options indicates that the prefetch or multifetch option is to be used for the command. The selected option is active if either the ISN sequence (I) or GET NEXT (N) option is specified in the command option 2 field. The multifetch/prefetch option can improve performance by eliminating the time needed for single-record fetches. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.
R	(return) returns response code 145 if the record to be read and held by an L4 command is not available.

Command Option 2

Option	Description
F	(first unused ISN) returns the next higher unused ISN for the specified file in the ISN field. The "next unused ISN" is determined by referring to the file control block (FCB). Do not use the "F" option when reading Adabas expanded files.
I	(read by ISN sequence) reads the record identified by the ISN specified in the ISN field if the ISN is present in the file. If the ISN is not present in the file, the record with the next higher ISN is read and that record's ISN is returned in the ISN field. If the ISN is not present and no higher ISN is present in the file, no record is read and response code 3 is returned.
N	(GET NEXT) reads the records identified by the ISNs in an ISN list without the user having to provide the ISN of the record to be read with each L1/L4 call. Adabas selects the ISN from the list and reads the record identified by that ISN. The ISN list to be used must be identified by the command ID field and must have been created previously by an Sx command. Response code 3 is returned when all the ISNs in the list have been selected. Refer to the section *ISN List Processing* for more information.

Additions 2: Length of Compressed and Decompressed Record

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

Note:
This length information is not returned when the prefetch feature is being used.

If the L1 or L4 command returns a nonzero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID

This field may be used to provide a separate format ID which is to be used to identify the internal format buffer used for this command, or to provide a global format ID.

If the high-order (leftmost) bit of the additions 5 field is not set to 1, the value provided in the command ID field is used as the format ID as well.

If the leftmost bit is set to 1, The rightmost four (fifth through eighth) bytes of the additions 5 field (additions 5 + 4(4)) will be used as the format ID.

If the two high-order (leftmost) bits of the first byte of additions 5 field are set to one (B’11’), the value in all eight bytes of the additions 5 field (additions 5 + 0(8)) is used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Buffers

Format Buffer

The user specifies the fields to be read in this buffer. format buffer syntax and examples are provided in the section *Adabas Calling Procedure*.

"C." in the first two positions of the format buffer causes the record to be returned in compressed instead of decompressed format.

Record Buffer

Adabas returns the requested field values in this buffer. All values are returned according to the standard format and length of the field unless the user specifies a different length and/or format in the format buffer.

ISN Buffer

The ISN buffer is used only if the command-level multifetch or prefetch option is used. See the section *Using the Multifetch/Prefetch Feature* for more information. When multifetching is used, the L1/L4 command returns descriptor elements, each up to 16 bytes long, in the ISN buffer (see the section *BT/ET Multifetch Processing*).

Examples

For the Adabas file definitions used in all the examples in this section, see *File Definitions Used in Examples*.

Example 1: Reading a Single Record

ISN 4 in file 1 is to be read. The values for fields AA and AB are to be returned.

Control Block

Command Code	L1	
Command ID	bbbb	only 1 record is to be read
File Number	1	
ISN	4	
Format Buffer Length	6	or larger
Record Buffer Length	10	or larger
Command Option 2	b	GET NEXT or read ISN sequence options not used
Additions 3	bbbbbbbb	file not security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	AA,AB

Example 2: Reading a Set of Records

A set of records for which the ISNs have been obtained previously by a find command are to be read from file 2. The values for fields RA and XB are to be returned with the value for field XB to be returned with length 3 and format U.

Control Block

Command Code	L1	
Command ID	ABCD	a nonblank CID is recommended for a series of read operations in which the same fields are being read in each record
File Number	2	
ISN	n	ISNs are taken from the ISN list created by the find command*
Format Buffer Length	10	or larger
Record Buffer Length	11	or larger
Command Option 2	b	GET NEXT or read ISN sequence options not used
Additions 3	password	file is security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	RA,XB,3,U

* n indicates an ISN from the ISN list which resulted from the find command. The L1 call is repeated for each ISN in the ISN list.

Example 3: Reading a Set of Records Using the GET NEXT Option

The requirement as stated for example 2 may also be satisfied by using the GET NEXT option.

Control Block

Command Code	L1	
Command ID	ABCD	CID of ISN list to be used
File Number	2	
ISN	0	the entire ISN list is to be selected starting with the first ISN in the list
Format Buffer Length	10	or larger
Record Buffer Length	11	or larger
Command Option 2	N	GET NEXT option to be used
Additions 3	password	file is security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	RA,XB,3,U

The L1 call is repeated for each ISN in the ISN list. No changes to the control block are required between L1 calls. Response code 3 will be returned when all the ISNs in the list have been selected.

Example 4: Read with Hold

ISN 5 in file 2 is to be read and held for updating. The values for fields XC and XD are to be returned.

Control Block

Command Code	L4	read with hold
Command ID	bbbb	only 1 record to be read
File Number	2	
ISN	5	
Format Buffer Length	6	or larger
Record Buffer Length	14	or larger
Command Option 1	b	response code 145 option not used
Command Option 2	b	GET NEXT or read ISN sequence options not used
Additions 3	password	file is security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	XC,XD

Example 5: Read Using the Read ISN Sequence Option

File 1 is to be read using the read ISN sequence option. The values for fields AA, AB, and AC are to be returned.

Control Block

Command Code	L1	
Command ID	BCDE	nonblank CID is recommended when a series of records for which the same fields are to be returned is to be read
File Number	1	
ISN	1	if ISN 1 is not present, the record of the next higher ISN in the file is read, and the ISN is returned in this field
Format Buffer Length	6	or larger
Record Buffer Length	30	or larger
Command Option 2	I	read ISN sequence option invoked
Additions 3	bbbbbbbb	file is not security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	GA,AC.

Adabas returns the ISN of the record which has been read in the ISN field of the control block. The record with the next higher ISN may be read by adding 1 to the ISN field and repeating the L1 command.

Example 6: Reading Multiple-Value Fields and Periodic Groups

The record identified by ISN 2 in file 1 is to be read. The value for field AA, all values for the multiple-value field MF, and all occurrences of the periodic group GB are to be returned.

Alternative 1: For Six or More Occurrences

1. Issue a L1 call to obtain the count of the number of values which exist for MF, and the highest occurrence count for GB.

   Control Block Field

Command Code	L1	
Command ID	bbbb	only 1 record is to be read
File Number	1	
ISN	2	
Format Buffer Length	8	or larger
Record Buffer Length	2	or larger
Command Option 2	b	GET NEXT or read ISN sequence option not used

   Buffer Areas

Format Buffer	MFC,GBC

2. Assuming that the result of the above L1 call was the record containing four values for MF and six occurrences for GB, repeat the L1 call using the following format buffer:

   ```
   AA,MF01-04,GB01-06
   ```

   For each call, the length of the format and record buffers must be large enough to hold all entries and values.

   When using this procedure to read a series of records, a valid command ID should be used in step 1, and no command ID (blanks) should be used for step 2 since the content of the format buffer may vary with each step 2 call.

Alternative 2: For Fewer Than Six Occurrences

An alternative solution for example 5 usually provides better performance if the number of values/occurrences is small (less than six) in a large percentage of the records.

1. Issue a L1 call in which the counts for MF and GB are requested, plus the expected number of values and occurrences of MF and GB, plus field AA.

   Assuming that the expected number of values for MF is 2, and the expected number of occurrences of GB is 3, the format buffer for step 1 would be

   ```
   AA,MFC,GBC,MF01-02,GB01-03.*
   ```

   * Maximum performance is normally achieved if a number which will retrieve all of the values/occurrences in 90 per cent of the records is specified.

2. If either the count received for MF exceeds 2 or the count received for GB exceeds 3, a format buffer similar to that in step 2 of the previous example could be used to obtain the additional values and/or occurrences. Otherwise, no additional call is required.

L2 / L5 Command: Read Physical Sequential

The L2/L5 commands read records in the sequence in which they are physically located in Data Storage.

This chapter covers the following topics:

- Function and Use
- Command: L2 / L5
- Control Block
- Buffers
- Additional Considerations
- Examples

Function and Use

The L5 command performs the same function as the L2 command, and, in addition, places each record read in hold status. If the record to be read and held is currently being held by another user, the user will be placed in wait status until the record becomes available. If the L5 command was issued with the command option 1 field set to "R", Adabas returns response code 145 if the record is not available.

The L2/L5 commands do not read records in any particular logical order unless the records were loaded initially in a particular logical sequence and no subsequent update changed this order.

The L2/L5 commands may be used to read an entire file at optimum speed since no access is required to the Associator (as with the L3 command), and all physical blocks are read in consecutive sequence.

The user specifies the file to be read and the fields within each record for which values are to be returned. The fields are specified in the format buffer. Adabas returns the requested field values in the record buffer.

The multifetch/prefetch option allows prior accessing of one or more sequential records, reducing overall operating time and eliminating the time needed for single-record fetches. Multi-/prefetching can be enabled by specifying "M", "O" (for multifetching), or "P" (for prefetching) in the command option 1 field. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

The compressed option (set by specifying "C." in the format buffer) causes the record read to be returned in compressed format, that is, as stored internally by Adabas.

Command: L2 / L5

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	A
ISN LOWER LIMIT	17-20	binary	F	A
	21-24	--	--	--
FORMAT BUFFER LENGTH	25-26	binary	F	U
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-32	--	--	--
ISN BUFFER LENGTH *	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
	36-44	--	--	--
ADDITIONS 2	45-48	binary / binary	--	A
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
ADDITIONS 5	65-72	alphanumeric	F	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	F**	U
RECORD BUFFER	--	A
ISN BUFFER *	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	The ISN buffer and length required only of the multi-/prefetch option is specified
**	May contain compress option control characters "C."
--	Not used

Control Block

Command Code

L2/L5

Command ID

This field must be set to a non-blank, non-zero value. It is used by Adabas to provide the records in the correct physical order and to avoid the repetitive interpretation of the format buffer. The value provided must not be modified during any given sequential pass of a file.

The first byte of this field may not be set to hexadecimal ’FF’.

If the command ID value is X’FFFFFFFF’, automatic command ID generation will be in effect. In this case, the Adabas nucleus will generate values for command ID beginning with X’00000001’, and will increment the value by 1 for each L2/L5 call. When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the additions 5 field for separate format ID or global format ID usage.

File Number

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

Response code 3 indicates an end-of-file (EOF) condition was detected.

ISN

If this field is set to zero before the initial L2/L5 call, the sequential pass will begin with the first record contained in the first physical block of the file.

If this field is set to an ISN value before the initial L2/L5 call, the sequential pass will begin at the first record physically located after the record identified by the ISN specified. The ISN specified must be present in the file.

This field need not be modified by the user after the initial L2/L5 call. Adabas returns the ISN of the record which has been read in this field.

ISN Lower Limit: Multifetch Record Count

If either "M" or "O" (multifetching option) is specified in the command option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and ISN buffer lengths. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

Format Buffer Length

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified. If either multifetch option "M" or "O" is specified in the command option 1 field, this value must be less than 32 kilobytes.

Record Buffer Length

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified. If either multifetch option "M" or "O" is specified in the command option 1 field, this value must be less than 32 kilobytes.

ISN Buffer Length: Only with Command-Level Multifetch/Prefetch Option

The ISN buffer length (in bytes). The ISN buffer area defined in the user program must be as large as (or larger than) the length specified.

Command Option 1

To improve performance by eliminating the time needed for single-record fetches, set option M, O, or P to enable multifetch or prefetch processing for the command. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

Option	Description
M	Enable multifetching
O	Use multifetching with the "R" option described below
P	Use prefetching
R	(return) returns response code 145 if the record to be read and held by an L5 command is not available.

Additions 2: Length of Compressed and Decompressed Record

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

Note:
This length information is not returned when the prefetch feature is being used.

If the L2 or L5 command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Additions 3: Password

This field is used to provide an Adabas security or Adabas SAF Security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID

This field may be used to provide a separate format ID which is to be used to identify the internal format buffer used for this command, or to provide a global format ID.

As long as the high-order (leftmost) bit of the first byte of the additions 5 field is not set to 1, the value provided in the command ID field will be used as the format ID as well.

If, however, this high-order bit is set to 1, the fifth through eighth bytes of the additions 5 field (additions 5 + 4(4)) will be used as the format ID.

If the two high-order (leftmost) bits of the first byte of additions 5 field are set to one (B’11’), the value in all eight bytes of the additions 5 field (additions 5 + 0(8)) is used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Buffers

Format Buffer

The fields for which values are to be returned are specified in this buffer. Format buffer syntax and examples are provided in the section *Adabas Calling Procedure*.

"C." in the first two positions of the format buffer indicates that the compressed option is to be in effect. This causes the record to be returned in compressed instead of decompressed format.

The format buffer may not be modified after the initial L2/L5 call has been issued.

Record Buffer

Adabas returns the requested field values in this buffer. All field values are returned according to the standard length and format of each field, unless the user specifies a different length and/or format in the format buffer.

ISN Buffer

The ISN buffer is used only if the command-level multifetch or prefetch option is used. See the section *Using the Multifetch/Prefetch Feature* for more information. When multifetching is used, the L2/L5 command returns record descriptor elements, each up to 16 bytes long, in the ISN buffer (see the section *BT/ET Multifetch Processing*).

Additional Considerations

The following additional considerations are applicable for the L2/L5 command:

1. The command ID used with the L2/L5 command is saved internally and used by Adabas. It is released by Adabas when an end-of-file condition is detected, an RC or CL command is issued, or the Adabas session is terminated. The same command ID may not be used by the user for another read sequential command until it has been released.

2. The user is permitted to update and/or delete records from a file that is being read by the user with an L2/L5 command. Adabas maintains information about the last and next record to be provided to the user, and is able to provide the correct next record despite any interim record update or deletion performed by the user.

3. If another user is updating the file being read with an L2/L5 command, it is possible that the user reading with the L2/L5 command will not receive one or more records in the file, or may receive the same record more than once during a given sequential pass of the file.

Examples

Example 1:

File 2 is to be read in physical sequential order. All the values for all the fields within each record are to be returned.

Control Block

Command Code	L2	
Command ID	EXL2	nonblank CID required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	3	or larger
Record Buffer Length	49	or larger
Command Option 1	b	return option not used
Additions 3	password	file 2 is security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	RG

The L2 call is repeated to obtain each successive record. The ISN field need not be modified between calls.

Example 2:

File 2 is to be read in physical sequential order. The values for fields RA, XA, and XB (3 bytes unpacked) are to be returned. Each record read is to be placed in hold status for updating purposes.

Control Block

Command Code	L5	
Command ID	EXL5	nonblank CID is required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	13	or larger
Record Buffer Length	21	or larger
Command Option 1	b	response code 145 option not used
Additions 3	password	file 2 is security-protected
Additions 4	bbbbbbbb	file 2 is not ciphered

Buffer Areas

Format Buffer	RA,XA,XB,3,U

The L5 call is repeated to obtain each successive record. The ISN field need not be modified between L5 calls.

To complete logical transactions and release held records, ET logic users should issue an ET command. Non-ET users should release held records with an A4, E4, or RI command.

L3 / L6 Command: Read Logical Sequential

The L3/L6 command is used to read afile in logical sequential order based on the sequence of the values for a given descriptor.

This chapter covers the following topics:

- Function and Use
- Command: L3 / L6
- Control Block
- Buffers
- Additional Considerations
- Examples

Function and Use

The user specifies the

- file to be read;
- search descriptor, whose values are to control the read sequence (phonetic descriptors and descriptors contained within or derived from a field within a periodic group may not be used);
- field or fields within each record for which values are to be returned.

Each L3/L6 command returns the requested values for one record’s field or fields in the record buffer. Repeating the command returns the values of each field in order of the search descriptor values.

The ascending and descending options specify whether the records are read in ascending or descending order. The ascending, descending, and value start options may be used to position to a given value before sequential reading begins or to position to a given value during sequential reading. This allows the user to position to a specific record in the file without having to read each record.

The multifetch/prefetch option allows prior accessing of one or more sequential records, reducing overall operating time and eliminating the time needed for single-record fetches. Multi-/prefetching can be enabled by specifying "M", "O" (for multifetching) or "P" (for prefetching) in the command option 1 field. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

If the format buffer contains "C.", the field values for each record read are returned in compressed format; otherwise, in uncompressed format.

Processing for the L3 command uses an internal table (the table of sequential commands) to keep track of the most recently returned value and ISN. "Normal" and "expanded" files are operationally diverse and for performance reasons, the value and ISN for each are handled differently in the table.

Specifically, the entries for "normal" files keep the value and ISN that *was returned* on the last call. The entries for "expanded" files keep the value and ISN that must be returned to the application on the *next* call.

This means that for "normal" files where concurrent updates are taking place, the L3 command returns a newly inserted value/ISN combination as long as it is *greater than or equal to* the last value/ISN returned.

For "expanded" files where concurrent updates are taking place, the L3 command returns a newly inserted value/ISN combination as long as it is *greater than or equal to* the value/ISN that is kept in the table of sequential commands entry as the *next* value/ISN to be returned.

The L6 command performs the same function as the L3 command and, in addition, places each record read in hold status. If a record to be read and held is currently being held by another user, the read operation stops and the user is placed in wait status until the record becomes available or an operation timeout occurs. If the command option 1 field is set to either "O" or "R" and a requested record is already being held, the L6 command returns response code 145.

Repositioning to a Particular Value

See the section *ISN: Optional Start ISN/Record Read ISN* in section *L3 / L6 Control Block* for information about specifying a starting value for the sequence-controlling descriptor. The following procedure is used to *reposition* to a particular value during a sequential pass:

1. Set the last six positions of the additions 1 field (the descriptor controlling the read sequence) to blanks.

2. Insert in the value buffer the value to which the read sequence is to reposition.

3. Set the ISN field to zero.

4. Set the command option 2 field to "A", "D", or "V".

Changing the Direction of the Read

Within a logical read sequence, the direction of the read can be changed at any time from ascending to descending or back without repositioning by specifying "D" or "A" for command option 2.

This option is not supported with prefetch or multifetch.

Command: L3 / L6

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	binary	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	A
ISN LOWER LIMIT	17-20	binary	F	A
	21-24	--	--	--
FORMAT BUFFER LENGTH	25-26	binary	F	U
RECORD BUFFER LENGTH	27-28	binary	F	U
SEARCH BUFFER LENGTH	29-30	binary	F *	U
VALUE BUFFER LENGTH	31-32	binary	F *	U
ISN BUFFER LENGTH **	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	A
ADDITIONS 2	45-48	binary / binary	--	A
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
ADDITIONS 5	65-72	alphanumeric	F	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	F ***	U
RECORD BUFFER	--	A
SEARCH BUFFER *	F	U
VALUE BUFFER *	F	U
ISN BUFFER **	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Required only if value start option is being used
**	The ISN buffer and length required only if the multi-/prefetch option is specified
***	May contain compress option control characters "C."
--	Not used

Control Block

Command Code

L3/L6

Command ID

This field must be set to a non-blank, non-zero value. The value provided is used by Adabas to provide the records in the correct sequence and to avoid repetitive interpretation of the format buffer.

The first byte of this field may not be set to hexadecimal ’FF’, and must not be changed during a given sequential pass of a file.

If the command ID value is X’FFFFFFFF’, the command ID is automatically generated. The Adabas nucleus generates command ID values beginning with X’00000001’, and increments the value by 1 for each L3/L6 call.

When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the additions 5 field for separate format ID and/or global format ID usage.

File Number

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

Response code 3 indicates that an end-of-file condition has been detected.

ISN: Optional Start ISN/Record Read ISN

For initial or subsequent positioning to a specific location in the logical read sequence when the last six bytes of the additions 1 field are set to blanks, the ISN field is used together with the starting descriptor value specified in the value buffer to determine the position where reading starts or restarts.

If an existing read sequence is to be continued without repositioning when the last six bytes of the additions 1 field are not changed from the most recent read command of the sequence, the ISN field is ignored.

Adabas returns in the ISN field the ISN of the record that was read.

When positioning or repositioning in the logical read sequence occurs, the ISN field is not used for positioning if the comparator for the starting descriptor value is set to

- GT (greater than) for an ascending read sequence; or
- LT (less than) for a descending read sequence.

Otherwise, the ISN field plays a role and the following rules apply:

Rules for Reading Forward (Ascending Option)

If the starting descriptor value specified in the value buffer *is* present in the logical read sequence, specifying

- an ISN of zero results in positioning to the *first* (*lowest*) ISN with that descriptor value;
- a non-zero ISN results in positioning to the *lowest* ISN *greater than* the specified ISN with that descriptor value, if such an ISN exists. Otherwise, positioning is to the *first* ISN of the *next higher* descriptor value.

If the starting descriptor value specified in the value buffer is *not* present in the logical read sequence, positioning is always to the *first* ISN of the *next higher* descriptor value, regardless of the specified ISN.

Rules for Reading Backward (Descending Option)

If the starting descriptor value specified in the value buffer *is* present in the logical read sequence, specifying

- an ISN of zero results in positioning to the *last* (*highest*) ISN with that descriptor value;
- a non-zero ISN results in positioning to the *highest* ISN *less than* the specified ISN with that descriptor value, if such an ISN exists. Otherwise, positioning is to the *last* ISN of the *next lower* descriptor value.

If the starting descriptor value specified in the value buffer is *not* present in the logical read sequence, positioning is always to the *last* ISN of the *next lower* descriptor value, regardless of the specified ISN.

ISN Lower Limit: Multifetch Record Count

If either "M" or "O" (multifetching option) is specified in the command option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and ISN buffer lengths. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

Format Buffer Length

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified.

Record Buffer Length

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified.

Search Buffer Length

If the search buffer is used, the length of the search buffer must be set in this field. The search buffer is required if the value start option is used. If the search buffer is not used, the value zero (0) must be set in this field.

Value Buffer Length

If the value buffer is used, the length of the value buffer must be set in this field. The value buffer is required if the value start option is used. If the value buffer is not used, the value zero (0) must be set in this field.

ISN Buffer Length: Only with Command-Level Multi-/Prefetch Option

The ISN buffer length (in bytes). The ISN buffer area defined in the user program must be at least as large as the length specified.

Command Option 1

Option	Description
M	(multifetching)
O	(multifetching with the "R" option)
P	(prefetching)
	Specifying one of these options indicates that the (command-level) prefetch or multifetch option is to be used. The multifetch/prefetch option can improve performance by allowing prior access to one or more sequential records, reducing overall operating time and eliminating the time needed for single-record fetches. For more information, see the section *Using the Multifetch/Prefetch Feature*.
R	(return) returns response code 145 if the record to be read and held by an L6 command is not available.

Command Option 2

This option specifies the sequence in which the descriptor’s entries are to be processed.

Option	Description
A	(ascending) processes entries for the descriptor in ascending sequence with an optional specification of a starting value. A starting descriptor value can be specified in the value buffer. In addition, a non-null value in the ISN field can be used to position within multiple matching descriptor value entries. If the search buffer and value buffer are omitted (SBL and VBL are set to zero (0)), all entries for the descriptor are processed.
D	(descending) processes entries for the descriptor in descending sequence with an optional specification of a starting value. A starting descriptor value can be specified in the value buffer. In addition, a non-null value in the ISN field can be used to position within multiple matching descriptor value entries. If the search buffer and value buffer are omitted (SBL and VBL are set to zero (0)), all entries for the descriptor are processed.
V	(value start) specifies an *ascending* sequential pass to begin (or continue) at a user-specified value rather than with the lowest (or next) value of the descriptor. The user-specified value *must* be in the value buffer, and the value’s length and format *must* be specified in the search buffer. A non-null value specified in the ISN field can further qualify the read operation. *This option was used in Adabas prior to version 6; it is maintained to support existing programs.*
blank	specifies an *ascending* sequential pass without a starting value; all values present are processed. The search buffer and value buffer are ignored when present. *This option was used in Adabas prior to version 6; it is maintained to support existing programs.*

Additions 1: Descriptor Used for Sequence Control

The descriptor to be used to control the read sequence must be specified in this field. A descriptor, subdescriptor, or superdescriptor may be specified.

- Phonetic descriptors and descriptors contained within or derived from a field contained within a periodic group may not be specified.
- A descriptor which is a multiple-value field may be specified, in which case the same record may be read several times (once for each different value within a given record) during the sequential pass through the file.
- If the descriptor specified is defined with the null value suppression (NU) option, any records containing a null value for the descriptor will not be read.
- The descriptor name is specified in the first two positions of this field. All remaining positions must be set to blanks on the initial call.

The additions 1 field should not be modified between L3/L6 calls. The exception is when the user wishes to reposition to a particular value during a sequential pass. This is done by setting the last six positions of this field to blanks, inserting the value to which repositioning is to occur in the value buffer, setting the ISN field to zero, and setting the command option 2 field to "A", "D", or "V".

Additions 2: Length of Compressed and Decompressed Record

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

Note:
This length information is not returned when the prefetch feature is being used.

If the L3 or L6 command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are described in the *Adabas Messages and Codes* documentation.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the additions 5 field is set to 0, the value provided in the command ID field is used as the format ID as well.

If the leftmost bit is set to 1, the contents of the fifth through eighth bytes of the additions 5 field (additions 5 + 4(4)) are interpreted as the format ID.

If the two high-order (leftmost) bits of the first byte of additions 5 field are set to one (B’11’), the value in all eight bytes of the additions 5 field (additions 5 + 0(8)) is used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Buffers

Format Buffer

The fields for which values are to be returned must be specified in this buffer. The syntax and examples of format buffer construction are described in the *Adabas Calling Procedure*.

Specifying "C." in the first two positions indicates that the compressed option is in effect. This causes the record to be returned in compressed instead of decompressed format.

Record Buffer

Adabas returns the requested field values in this buffer. The field values are returned according to the standard format and length of each field, unless the user has requested a different format and/or length in the format buffer.

Search Buffer

Note:
Commas separate elements in the search buffer; a period terminates the syntax statement. For more information about search buffer syntax, see the discussion beginning on page .

The search buffer length (SBL) must be set in the control block. If the search buffer is not used, SBL must be set to zero (0).

If the value start (V) option is used for command option 2, the starting value of the descriptor used for sequence control *must* be specified in the value buffer and the value's length and format must be specified in the search buffer.

If the ascending (A) or descending (D) option is used for command option 2, a search buffer may be used but is not required. If either the search or value buffer is omitted, all values for a given descriptor are processed. If a starting value, ending value, or both are specified in the value buffer, the search buffer *must* be used to limit the number of descriptor entries retrieved.

The length and format of the descriptor value as provided in the value buffer must be specified in the search buffer if different from the standard length and/or format of the named descriptor.

When a single value is provided in the value buffer (that is, the starting value with option A or the ending value with option D), the syntax for the search buffer is

```
name [ ,length ] [ ,format ] [ ,comparator ]
```

When two values are provided in the value buffer (that is, a starting and an ending value), the syntax for the search buffer is

```
name [ ,length ] [ ,format ] ,S ,name [ ,length ] [ ,format ]
```

The elements used in the search buffer syntax statements are

name	is the name of the descriptor to be used for sequence control. The name specified must be the same as that specified in the additions 1 field.	
length	is the length of the value provided in the value buffer. If the length is not specified, it is assumed that the value is being provided using the standard length of the descriptor. See the section *Length and Data Format* for the allowed length settings.	
format	is the format of the value provided in the value buffer. If the format is not specified, it is assumed that the value is being provided using the standard format of the descriptor.See the section *Length and Data Format* for the allowed format settings.	
comparator	identifies the scope of the read sequence:	
	GE	greater than or equal to the value to/from the highest value (the default)
	GT	greater than the value to/from the highest value
	LE	less than or equal to the value to/from the lowest value
	LT	less than the value to/from the lowest value
S	A FROM-TO range (inclusive) that involves two search expressions. The same descriptor must be used in both expressions:	
	AA,S,AA.	valid
	AA,S,AB.	invalid

See *Example 3: Overview of Sequence Options* for an overview of sequence options resulting from the choice of A or D for command option 2 and various search and value buffer options.

Value Buffer

The value buffer length (VBL) must be set in the control block. If the value buffer is not used, VBL must be set to zero (0).

If the value start option is to be used, or if value repositioning is to be performed, the value with which reading is to start (or continue) must be specified in this buffer.

If the ascending (A) or descending (D) option is used for command option 2, a value buffer may be used but is not required. If either the search or value buffer is omitted, all values for a given descriptor are processed. If a starting value, ending value, or both are specified in the value buffer, the search buffer is required in order to limit the number of descriptor entries retrieved.

If the ascending (A) option is used for command option 2, reading starts (or continues) with the first record containing the value specified. If there are no records with keys equal to a start or reposition (continue) value, reading begins with the first record containing the next higher value.

When two values are provided in the value buffer, the first specifies the lower limit of a range and the second specifies the upper limit. Each value is either the starting or ending value depending on the command option 2 setting of A or D.

In addition, if the ISN field is set to a non-zero value, reading will start (or continue) or end with the first record whose ISN is present for the value specified, provided that the ISN is greater than the ISN in the ISN field. If no such ISN exists, the first record with the next higher (or lower) value is read.

Search and Value Buffer Examples

Example 1: ASCENDING Option with Optional Search and Value Buffer

Inverted list for the descriptor used for sequence control:

Value	ISN List
A	1, 4
B	2
D	3, 5

Initial L3/L6 command with command option 2 set to "A" (ascending option), or subsequent L3/L6 command with command option 2 set to "A" and the last six bytes of additions 1 reset to blanks:

User-Supplied Value	User-Supplied ISN	ISN Where Reading Starts or Continues
A	0	1
A	1	4
A	2	4
A	4	2
A	5	2
B	0	2
B	1	2
B	2	3
B	3	3
BABC	1	3
C	0	3
D	0	3
D	3	5
D	4	5
D	5	response code 3
E-Z	-	response code 3

Example 2: DESCENDING Option with Optional Search and Value Buffer

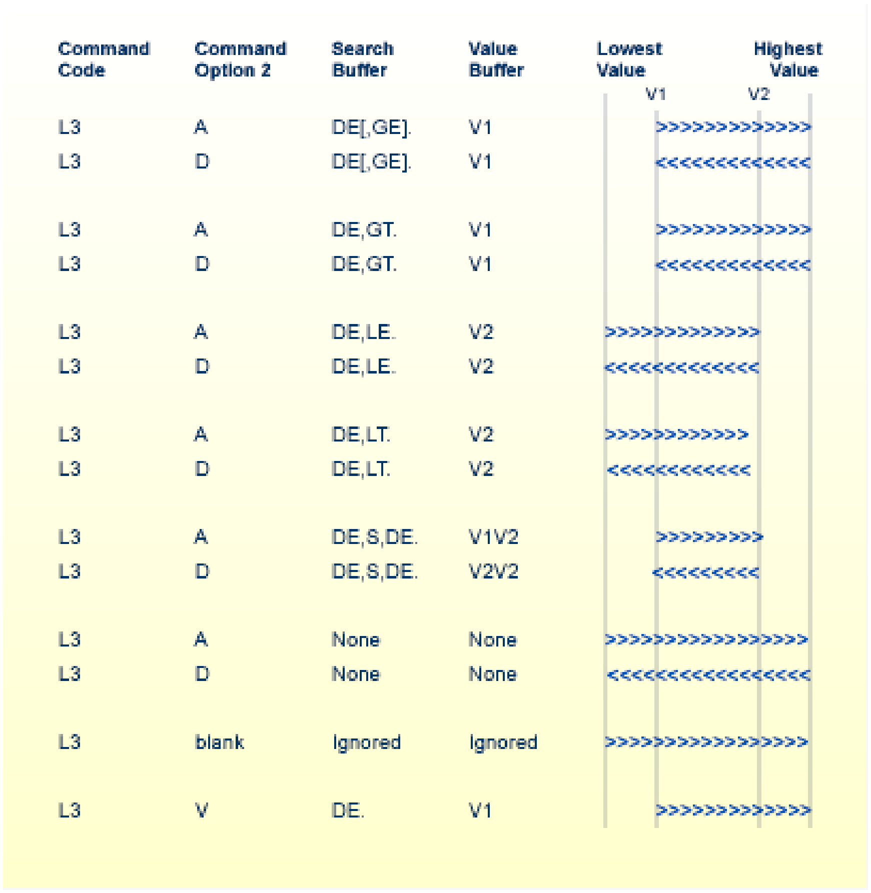
Inverted list for the descriptor used for sequence control:

Value	ISN List
A	1, 9, 25
B	3, 18, 21
C	7, 8, 11

Initial L3/L6 command with command option 2 set to "D" (descending option), ISN=0 for optional start, search buffer comparator set to "LT", and value buffer containing value "C". The start position is the *highest* ISN of the next *lower* descriptor value: in this example, ISN 21 of descriptor value "B".

Example 3: Overview of Sequence Options

The following table illustrates the possibilities for using the ascending/descending option in conjunction with various search buffer and value buffer contents. The following applies to the file being read and ISN=0 (no further positioning within matching start values):



Command Code	Command Option 2	Search Buffer	Value Buffer	Lowest Value V1 V2 Highest Value
L3	A	DE[,GE].	V1	>>>>>>>>>>>>>
L3	D	DE[,GE].	V1	<<<<<<<<<<<<<
L3	A	DE,GT.	V1	>>>>>>>>>>>>>
L3	D	DE,GT.	V1	<<<<<<<<<<<<<
L3	A	DE,LE.	V2	>>>>>>>>>>>>>
L3	D	DE,LE.	V2	<<<<<<<<<<<<<
L3	A	DE,LT.	V2	>>>>>>>>>>>>
L3	D	DE,LT.	V2	<<<<<<<<<<<<
L3	A	DE,S,DE.	V1V2	>>>>>>>>>
L3	D	DE,S,DE.	V2V2	<<<<<<<<<
L3	A	None	None	>>>>>>>>>>>>>>>>>
L3	D	None	None	<<<<<<<<<<<<<<<<<
L3	blank	Ignored	Ignored	>>>>>>>>>>>>>>>>>
L3	V	DE.	V1	>>>>>>>>>>>>>

Overview of Sequence Options

ISN Buffer

The ISN buffer is used only if the command-level multifetch or prefetch option is used. See the section *Using the Multifetch/Prefetch Feature* for more information.

Additional Considerations

The following additional considerations are applicable for the L3/L6 command:

1. The command ID used with the L3/L6 command is saved internally and used by Adabas. It is released by Adabas when an end-of-file condition is detected, an RC or CL command is issued, or the Adabas session is terminated. The same command ID may not be used by the user for another read sequential command until the command ID has been released.

2. The user is permitted to update and/or delete records from a file that is being read by the user with an L3/L6 command. If the inserted record or records are to be read, the user must reposition after having inserted the record or records.

3. If any user is updating the file being read with an L3/L6 command, it is possible that the user reading with the L3/L6 command either may not receive one or more records in the file, or may receive the same record more than once during a sequential pass of the file.

Note:
You should avoid updating any descriptor field that controls the read sequence with a value greater than that which it currently contains.

Examples

Example 1

File 2 is to be read in logical sequential order. The descriptor RB is to be used for sequence control. The values for the fields RA and RB are to be returned. The entire file is to be read.

Control Block

Command Code	L3	
Command ID	EX01	a non-blank CID is required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger
Search Buffer Length	0	
Value Buffer Length	0	
Command Option 1	b	response code 145 or prefetch options not used
Command Option 2	A	Ascending order with no search or value buffer specified
Additions 1	RBbbbbbb	descriptor RB to be used for sequence control
Additions 3	password	file is security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	RA,RB.

The L3 call is repeated to obtain each successive record. The CID and additions 1 field must not be modified between L3 calls. Response code 3 will be returned when all records have been read.

Example 2: Read Logical Sequential with Record Hold

File 1 is to be read in logical sequential order. The descriptor AA is to be used for sequence control. The values for fields AA and AB are to be returned. Each record read is to be placed in hold status.

Control Block

Command Code	L6	
Command ID	EX02	non-blank CID required
File Number	1	
ISN	0	all records to be read
Format Buffer Length	3	or larger
Record Buffer Length	10	or larger
Search Buffer Length	0	
Value Buffer Length	0	
Command Option 1	b	response code 145 or prefetch options not used
Command Option 2	b	Ascending order with no search or value buffer specified
Additions 1	AAbbbbbb	descriptor AA to be used for sequence control
Additions 3	bbbbbbbb	file is not security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	GA.

To complete logical transactions and release held records, ET logic users should issue an ET command. Non-ET users should release held records with an A4, E4, or RI command.

Example 3: ASCENDING Option

The same requirement as example 1 except that reading is to begin with the value "N".

Control Block

Command Code	L3	
Command ID	EX03	non-blank CID required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger
Search Buffer Length	7	or larger
Value Buffer Length	1	or larger
Command Option 1	b	response code 145 option not used
Command Option 2	A	ascending option with start value supplied
Additions 1	RBbbbbbb	RB is to be used for sequence control
Additions 3	password	file is security-protected
Additions 4	b	file is not ciphered

Buffer Areas

Format Buffer	RA,RB.	
Search Buffer	RB,1,A.	
Value Buffer	N	reading to begin with value N

The initial L3 call will result in the return of the first record which contains the value "N" for the descriptor RB. If no records exist with this value, the first record which contains a value greater than "N" (such as "NA") will be returned.

Example 4: DESCENDING Option with Repositioning

The same requirement as example 3 except that a value repositioning is to be performed. The sequential read process is to continue with the value "Q".

Control Block

Command Code	L3	
Command ID	EX03	the CID field may not be changed when repositioning
File Number	2	
ISN	0	this field must be set to zeros for value repositioning
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger
Search Buffer Length	7	or larger
Value Buffer Length	1	or larger
Command Option 1	b	response code 145 option not used
Command Option 2	D	descending option with value repositioning
Additions 1	RBbbbbbb	the last six positions of this field must be set to blanks for value repositioning
Additions 3	password	file 2 is security-protected
Additions 4	bbbbbbbb	file 2 is not ciphered

Buffer Areas

Format Buffer	RA,RB.	
Search Buffer	RB,1,A,LE.	
Value Buffer	Q	reposition to value "Q"

L9 Command: Read Descriptor Values

The L9 command reads the values of a specified descriptor.

This chapter covers the following topics:

- Function and Use
- Command: L9
- Control Block
- Buffers
- Additional Considerations
- Examples

Function and Use

The L9 command is used to determine the range of values present for a descriptor and the number of records that contain each value.

The user specifies the file containing the descriptor, the descriptor for which the values are to be returned, and the value at which processing is to begin. Each L9 call returns the next value for the descriptor in the record buffer, and the count of records containing that value in the ISN quantity field. The values may be either positive or negative. Null values for descriptors defined with the null value suppression (NU) option are not returned.

The multifetch and prefetch options improve performance by reading several descriptor values at a time. Multi-/prefetching can be enabled by specifying "M" (for multifetching) or "P" (for prefetching) in the command option 1 field. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

The "I" option specified in the command option 2 field returns the ISNs of each value in the record buffer. The L9 command reads the Associator inverted lists only; no Data Storage access is required.

The "A" and "D" options in the command option 2 field specify whether the descriptor values are read in ascending or descending order, respectively.

See *Example 3: Overview of Sequence Options* for an overview of sequence options resulting from the choice of A or D for command option 2 and various search and value buffer options.

Changing the Direction of the Read

Within a logical read sequence, the direction of the read can be changed at any time from ascending to descending or back by specifying "D" or "A" for command option 2.

This option is not supported with prefetch or multifetch.

Command: L9

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN LOWER LIMIT	17-20	binary	F	A
ISN QUANTITY	21-24	binary	--	A
FORMAT BUFFER LENGTH	25-26	binary	F	U
RECORD BUFFER LENGTH	27-28	binary	F	U
SEARCH BUFFER LENGTH	29-30	binary	F	U
VALUE BUFFER LENGTH	31-32	binary	F	U
ISN BUFFER LENGTH *	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	U
ADDITIONS 2	45-48	binary	--	A
ADDITIONS 3	49-56	alphanumeric	F	A
	57-64	--	--	--
ADDITIONS 5	65-72	alphanumeric	F	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	F	U
RECORD BUFFER	--	A
SEARCH BUFFER	F	U
VALUE BUFFER	F	U
ISN BUFFER *	--	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	The ISN buffer and length required only if the multi-/prefetch option is specified
--	Not used

Control Block

Command Code

L9

Command ID

This field must be set to a non-blank, non-zero value. This value is used by Adabas to provide the values in the correct sequence and to avoid the repetitive interpretation of the format buffer.

This field must not be modified during a given sequential pass of the file.

The first byte of this field may not be set to hexadecimal ’FF’.

File Number

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

Response code 3 indicates that an end-of-file condition has been detected.

ISN: Periodic Group Occurrence

If the descriptor for which values are to be returned is contained within a periodic group, Adabas returns in this field the occurrence number in which the value being returned is located. The occurrence number is provided in binary format in the two low-order bytes.

If the prefetch option is specified, any occurrence for prefetched values is returned in the header preceding the value in the ISN buffer.

ISN Lower Limit: Lowest ISN in Record Buffer

Adabas returns values in this field as follows:

Cmd Op 1	**Cmd Op 2**	**The L9 command . . .**
blank	I	returns the ISNs for each value in the record buffer; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the returned ISN list in the ISN lower limit field.
P(refetch)	I	returns the first ISN in the record buffer and all prefetched descriptor values in the ISN buffer, preceded by a 16-byte header; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the last value prefetched in the ISN lower limit field.
M(ultifetch)	I	returns the group of multifetched records in the record buffer and a description of these records in the caller's ISN buffer; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the last value multifetched in the ISN lower limit field.

If command option 1 is set to "M" (multifetch option), you can set

- a non-zero value in the ISN lower limit field to limit the number of values to be multifetched.
- zero in the ISN lower limit field to multifetch all values.

Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

ISN Quantity: Record Count

Except when the "return ISNs" option (I) is specified, Adabas returns in this field the number of records containing the value returned in the record buffer.

If the prefetch option is specified, the count of prefetched values is in the header that precedes the value in the ISN buffer.

Format Buffer Length

The format buffer length (in bytes). The format buffer area defined in the user program must be at least as large as the length specified.

Record Buffer Length

The record buffer length (in bytes). The record buffer area defined in the user program must be at least as large as the length specified.

Search Buffer Length

The search buffer length (in bytes). The search buffer area defined in the user program must be at least as large as the length specified.

Value Buffer Length

The value buffer length (in bytes). The value buffer area defined in the user program must be at least as large as the length specified.

ISN Buffer Length: Only with Command-Level Multifetch/Prefetch Option

The ISN buffer length (in bytes).

When the command option 1 field specifies "P", the L9 command uses the ISN buffer to hold prefetched descriptor values. The ISN buffer must be large enough to hold the largest descriptor value plus a 16-byte header preceding each value. The actual ISN buffer area defined in the user program must be at least as large as the length specified.

Command Option 1: Command-Level Multifetch/Prefetch Option

Specifying one of these options indicates that the (command-level) prefetch or multifetch option is to be used. The multifetch/prefetch option can improve performance reading several descriptor values at a time thereby eliminating the time needed for single-record fetches. For more information, see the section *Using the Multifetch/Prefetch Feature*.

Option	Description
M	(multifetching) In conjunction with the "M" option, you can set ● a non-zero value in the ISN lower limit field to limit the number of values to be multifetched. ● zero in the ISN lower limit field to multifetch all values. The format of the ISN buffer data with the "M" option reflects the standard record descriptor data format.
P	(prefetching) The L9 command puts all prefetched descriptor values in the ISN buffer, preceded by a 16-byte header. The ISN buffer must be large enough to hold the largest descriptor value plus 16 bytes. The actual ISN buffer area defined in the user program must be at least as large as the ISN buffer length specified.

Command Option 2: Return ISNs Option

Option	Description
I	returns the ISNs for each value in the record buffer. The L9 command reads the Associator inverted lists only; no Data Storage access is required.
A	the descriptor’s entries are processed in ascending order.
D	the descriptor’s entries are processed in descending order.

Additions 1: Descriptor Name

If both the search and value buffer lengths are set to zero, a value in the additions 1 field is the name of the descriptor for which values are to be returned. The name must be the same as the descriptor name specified in the format buffer.

In this case, L9 processes all values for the specified descriptor from the beginning of the file.

The descriptor name is specified in the first two positions of this field. The remaining positions must be set to blanks.

Additions 2: Response Subcodes

If the L9 command returns a nonzero response code, the rightmost two bytes of this field may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 5: Format ID, Global Format ID

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the additions 5 field is set to 0, the value provided in the command ID field is used as the format ID as well.

If the leftmost bit is set to 1, the contents of bytes 5 through 8 of the additions 5 field (additions 5 + 4(4)) are interpreted as the format ID.

If the two high-order (leftmost) bits of the first byte of additions 5 field are set to one (B’11’), the value in all eight bytes of the additions 5 field (additions 5 + 0(8)) is used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Buffers

Format Buffer

Note:
Commas separate elements in the format buffer; a period terminates the syntax statement. For more information about format buffer syntax, see the discussion beginning on page .

The format in which the values are to be returned must be specified in this buffer.

The syntax of the format buffer for L9 operation is

```
name  [ ,length ]  [ ,format ] .
```

where:

name	is the name of the descriptor for which values are to be returned. A phonetic descriptor or hyperdiscriptor may not be specified. A collation descriptor may only be specified if the decode option has been specified in its user exit: the value returned for it is not the index but the original field value. Subdescriptors, superdescriptors and descriptors defined as a multiple-value field may be specified.
length	is the length in which the value is to be returned. If length is not specified, the value will be returned using the standard length of the descriptor.
format	is the format in which the value is to be returned. The format specified must be compatible with the standard format of the descriptor. If no format is specified, the value will be returned using the standard format of the descriptor.

Record Buffer

The descriptor value for the descriptor specified in the search and value buffers is returned in this field. A different value is provided with each L9 call. If the descriptor is defined with the null value suppression option, no value for the descriptor will be returned. If the command option 2 field specifies "I", Adabas returns a list of ISNs containing the requested value as well as the value itself in this buffer. The ISNs are provided in ascending sequence.

The descriptor value is provided as follows:

```
length  value  count  ISN-list
```

where:

length	is a one-byte binary value which is the length of the value being provided. If the value has a standard length according to the descriptor type, this field is zero.
value	is a descriptor value.
count	is the number of values present in the file for the specified descriptor. This number may be returned in one or two bytes. If in one byte, the format is X’cc’ where "cc" is the count; if in two bytes, the format is X’8ccc’ where "ccc" is the count.
ISN-list	If the command option 2 field specified "I", the rest of the record buffer contains ISNs of the records containing this value. One record buffer will contain only the ISNs from one NI (normal index) block. Therefore: ● the record buffer should be large enough to contain an entire NI block; and ● the same value may appear several times with ascending ISNs.

Search Buffer

Note:
Commas separate elements in the search buffer; a period terminates the syntax statement. For more information about search buffer syntax, see the discussion beginning on page .

If both the search and value buffer lengths are set to zero, a value in the additions 1 field is the name of the descriptor for which values are to be returned. In this case, L9 processes all values for the specified descriptor from the beginning of the file. The search and value buffers are not used.

If a starting value, ending value, or both are specified in the value buffer, the search buffer is required in order to limit the number of descriptor entries retrieved.

The length and format of the descriptor value as provided in the value buffer must be specified in the search buffer if different from the standard length and/or format of the named descriptor.

When a single value is provided in the value buffer (that is, the starting value when command option 2 is set to 'A' or the ending value when command option 2 is set to 'D'), the syntax for the search buffer is

```
name [ i ]  [ ,length ]  [ ,format ] [ ,comparator ]
```

When two values are provided in the value buffer, the syntax for the search buffer is

```
name [ i ]  [ ,length ] [ ,format ]  ,S  ,name  [ ,length ] [ ,format]
```

The elements used in the search buffer syntax statements are as follows:

name	is the name of the descriptor for which values are to be returned. The name must be the same as that specified in the format buffer.	
i	is a one- to three-digit occurrence number subscript appended to the descriptor name if the descriptor is contained within a periodic group and only values within that particular occurrence are to be returned.	
length	is the length of the value provided in the value buffer. If the length is not specified, it is assumed that the value is being provided using the standard length of the descriptor. See on page for the allowed length settings.	
format	is the format of the value provided in the value buffer. If the format is not specified, it is assumed that the value is being provided using the standard format of the descriptor. See on page for the allowed format settings.	
comparator	identifies the scope of the read sequence:	
	GE	greater than or equal to the value to/from the highest value (the default)
	GT	greater than the value to/from the highest value
	LE	less than or equal to the value to/from the lowest value
	LT	less than the value to/from the lowest value
S	A FROM-TO range (inclusive) that involves two search expressions. The same descriptor must be used in both expressions:	
	AA,S,AA.	valid
	AA,S,AB.	invalid

Value Buffer

If both the search and value buffer lengths are set to binary zeros, a value in the additions 1 field is the name of the descriptor for which values are to be returned. In this case, L9 processes all values for the specified descriptor in the sequence specified in the command option 2 field: from the beginning (A option) or end (D option) of the file. The search and value buffers are not used.

If a starting value, ending value, or both are specified in the value buffer, the search buffer is required in order to limit the number of descriptor entries retrieved.

If the search buffer comparator is GE or GT and a single value is provided in the value buffer, it represents the starting value when command option 2 is set to A or the ending value when command option 2 is set to D.

When two values are provided in the value buffer, the first specifies the lower limit of a range and the second specifies the upper limit. Each value is either the starting or ending value depending on the command option 2 setting of A or D.

If a value specified is not present in the file, Adabas finds the next higher or lower value, depending on the command option 2 setting of A or D. In this case, search buffer comparator LE is equivalent to LT and GE is equivalent to GT. If the value specified is not present and no higher (or lower) value is present, response code 3 is returned.

ISN Buffer

The information held in the ISN buffer following an L9 command results from specifying the M (multifetch) or P (prefetch) option in the command option 1 field. The format of the ISN buffer data depends on which option was specified.

Data Format for the Multifetch Option (M)

See section *READ (Lx) Multifetch Processing* for the record descriptor data format.

Data Format for the Prefetch Option (P)

The ISN buffer holds the optionally prefetched descriptor values, each preceded by a 16-byte header. The 16-byte header preceding each value has the following format:

Byte	Usage
1-2	length of descriptor (including this header)
3-4	nucleus response code
5-8	nucleus internal ID
9-12	periodic group occurrence (see the ISN field description)
13-16	record count (see the ISN quantity field description)

Additional Considerations

The following additional considerations are applicable for the L9 command:

1. The command ID used with the L9 command is saved internally and used by Adabas. It is released by Adabas when an end-of-file condition is detected, an RC or CL command is issued, or the Adabas session is terminated. Until it is released, the command ID may not be used for another command.

2. If another user is updating the file being read with an L9 command, it is possible that the user reading with the L9 command will not receive one or more values in the file.

3. 3. Records can be updated in or deleted from a file being read by an L9 command. Adabas attempts to keep track of the last and the next value to be provided to the L9 command, and to provide the correct next value despite any interim update or deletion. However, if the record about to be accessed by the L9 command changes for some reason (it is updated or deleted by another user, for example), the L9 command continues processing as though no change occurred. In other words, a record deleted just before its inverted list entry is accessed by the L9 command is still considered a valid entry by the L9 command.

4. 4. An internal format buffer used by an L9 command must have been created by a previous L9 command. Non-L9 commands cannot use internal format buffers created by L9 commands.

Examples

Example 1

All values for the descriptor RB in file 2 are to be returned.

Control Block

Command Code	L9	
Command ID	L901	a non-blank CID is required
File Number	2	
Format Buffer Length	3	or larger
Record Buffer Length	10	or larger
Search Buffer Length	5	or larger
Value Buffer Length	1	or larger
Additions 3	password	file 2 is security-protected

Buffer Areas

Format Buffer	RB.	the values are to be returned using standard length and format
Search Buffer	RB,1.	the values for descriptor RB are to be returned, and the starting value is being provided with standard format and length equal to 1
Value Buffer	b	processing is to begin with the first value for RB equal or greater than ’b’

Each successive L9 call will result in the return of the next value (values are provided in ascending order). The number of records containing the value is returned in the ISN quantity field.

Example 2

The values for descriptor AB in file 1 are to be returned. Only values which are equal to or greater than 20 are to be returned.

Control Block

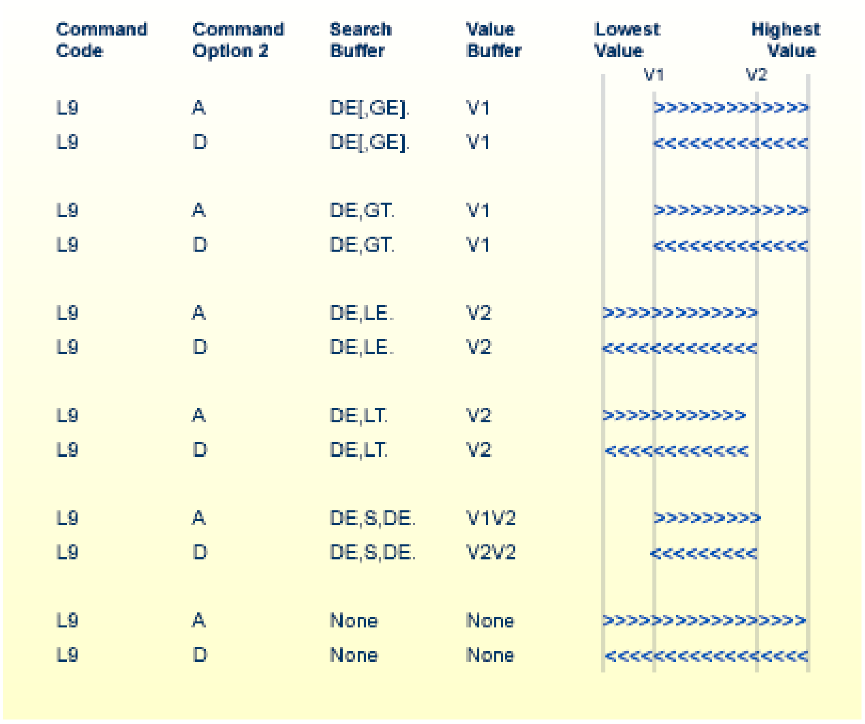
Command Code	L9	
Command ID	L902	a non-blank CID is required
File Number	1	
Format Buffer Length	7	or larger
Record Buffer Length	3	or larger
Search Buffer Length	7	or larger
Value Buffer Length	2	or larger
Additions 3	bbbbbbbb	file 1 is not security-protected

Buffer Areas

Format Buffer	AB,3,U.	the values are to be returned with length=3 and with format=unpacked
Search Buffer	AB,2,U.	the values for the descriptor AB are to be returned, and the starting value is to be provided as a 2-byte unpacked number
Value Buffer	X’F2F0’	processing is to begin with the first value for AB that is equal to or greater than 20

Example 3: Overview of Sequence Options

The following table illustrates the possibilities for using the ascending/descending option in conjunction with various search buffer and value buffer contents. The following applies to the file being read:



Command Code	Command Option 2	Search Buffer	Value Buffer	Lowest Value V1 V2 Highest Value
L9	A	DE[,GE].	V1	>>>>>>>>>>>>>
L9	D	DE[,GE].	V1	<<<<<<<<<<<<<
L9	A	DE,GT.	V1	>>>>>>>>>>>>>
L9	D	DE,GT.	V1	<<<<<<<<<<<<<
L9	A	DE,LE.	V2	>>>>>>>>>>>>>
L9	D	DE,LE.	V2	<<<<<<<<<<<<<
L9	A	DE,LT.	V2	>>>>>>>>>>>>
L9	D	DE,LT.	V2	<<<<<<<<<<<<
L9	A	DE,S,DE.	V1V2	>>>>>>>>>
L9	D	DE,S,DE.	V2V2	<<<<<<<<<
L9	A	None	None	>>>>>>>>>>>>>>>>>
L9	D	None	None	<<<<<<<<<<<<<<<<<

Overview of Sequence Options

LF Command: Read Field Definitions

The LF command reads the characteristics of all fields in a file.

This chapter covers the following topics:

- Function and Use
- Command: LF
- Control Block
- Record Buffer
- Example

Function and Use

The LF command is used to read the field definition information for a file. This command is used primarily by Adabas subsystems; it is normally not used by an application program.

The user specifies the file number for which the field definitions are to be returned.

Adabas provides the field information in the record buffer in one of three formats, according to the setting of the command option 2 field.

Command: LF

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
	5-8	--	--	--
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
	13-26	--	--	--
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-35	--	--	--
COMMAND OPTION 2	36	alphanumeric	F	U
	37-48	--	--	--
ADDITIONS 3	49-56	alphanumeric	F	A
	57-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	*	--
RECORD BUFFER	--	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block

Command Code

LF

File Number

The number of the file for which the field definition information is to be returned.

Note:
When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. If the LF command returns a nonzero response code, the rightmost two bytes of Adabas control block bytes 45-48 (additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Record Buffer Length

The record buffer length (in bytes). The length specified must be large enough to contain all field definition information for the file, but not larger than the size of the record buffer area defined in the user program. If you specify the internal format (I) command option, the maximum possible output is four Associator blocks.

Command Option 2: Type of Information to Be Displayed

The setting of the command option 2 field determines the format and type of field information to be returned in the record buffer.

Option	Description
S	returns all field information, including collation descriptor, subfield and superfield, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor information.
I	returns all field information in Adabas internal format.

If this field is left blank or contains binary zero, the LF command returns field information *excluding* sub-/super-/hyper-/phonetic or collation descriptor information. This is the same format as provided in Adabas version 4.

Additions 3: Password

This field is used to provide a security password. If the file to be used is not security-protected, this field should be set to blanks. If the file is security-protected, the user must provide a valid password.

If the accessed file is password-protected, Adabas replaces the password with blanks during command processing to protect password integrity.

Record Buffer

All field definition information is returned in the record buffer.

Command Option 2 is Set to "S"

If the command option 2 field is set to "S", all field information, including collation descriptor, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor and sub-/superfield information is returned in the following format:

Bytes	Usage
1-2	total length of information
3-4	number of fields in the FDT (including SDTs, as described below)
5-n	field definitions: each entry is 8 bytes, maximum number of entries is 926
(n+1) - m	special descriptor table (SDT) including ● sub-/superdescriptors (or sub-/superfields) ● phonetic descriptors ● hyperdescriptors ● collation descriptors The length of a sub-/phonetic/collation descriptor element is eight bytes. Superdescriptor or hyperdescriptor elements are two or more 8-byte entries long.

FDT Field Definitions

```
'F' field-name option level length format
```

Notation	Bytes	Usage
'F'	1	'F' indicates FDT field definition
field-name	2-3	field name

Notation	Bytes	Usage
option	4	definition options:
		bit 1=1 descriptor
		bit 2=1 fixed length
		bit 3=1 multiple-value field
		bit 4=1 null suppression
		bit 5=1 periodic-group field
		bit 6=1 parent of phonetic descriptor
		bit 7=1 parent of sub-/superdescriptor
		bit 8=1 unique descriptor
level	5	level number (in binary)
length	6	length

Notation	Bytes	Usage	
format	7	type of data:	
		A	alphanumeric
		B	binary
		F	fixed point
		G	floating point
		P	packed decimal
		U	unpacked decimal
		W	wide-character
	8	options (continued)	
		bit 1	(unused)
		bit 2=1	NV (not converted) option
		bit 3	(unused)
		bit 4=1	XI (exclude PE occurrence number from UQ) option
		bit 5=1	LA (long alpha) option
		bit 6	(unused)
		bit 7=1	NN option
		bit 8=1	NC option

Note:
A field within a periodic group has the following characteristics: - an option field (byte 4) with bit 5=1; and - a level field (byte 5) with level number *greater than* 1. The periodic group field itself always has option bit 5=1 and a level number of 1.

SDT Field Definitions

```
X  SDT-definition
```

where X is one of the following:

’C’	collation descriptor, see the section *Collation Descriptor Definition*
’H’	hyperdescriptor, see the section *Hyperdescriptor Definition*
’P’	phonetic descriptor, see the section *Phonetic Descriptor Definition*
’S’	subfield/descriptor, see the section *Subfield/Subdescriptor Definition*
’T’	superfield/descriptor, see the section*Superdescriptor/Superfield Definition*
X’00’	element continuation

Collation Descriptor Definition

```
’C’name option exit length p-field-name
```

Notation	Bytes	Usage
’C’	1	’C’ indicates collation descriptor
name	2-3	collation descriptor name
option	4	definition options:
		bit 1=1 descriptor
		bit 2=1 exclude PE occurrence number from UQ
		bit 3=1 multiple-value format
		bit 4=1 null-value suppression
		bit 5=1 periodic-group field
		bits 6-7 (unused)
		bit 8=1 unique descriptor
exit	5	collation exit number (binary, values 1-8 permitted)
length	6	length
p-field-name	7-8	parent-field name

Hyperdescriptor Definition

```
’H’name option exit length format X’00’
X’00’ X’00’ p-fieldname-list ...
```

Notation	Bytes	Usage	
’H’	1	’H’ indicates a hyperdescriptor definition	
name	2-3	hyperdescriptor name	
option	4	definition options:	
		bit 1	(unused)
		bit 2=1	fixed length
		bit 3=1	multiple value
		bit 4=1	null-value suppression
		bit 5=1	periodic group
		bit 6-7	(unused)
		bit 8=1	unique descriptor
level	5	hyperexit number (binary; values 1-31 permitted)	
length	6	length	
format	7	format:	
		A	alphanumeric
		B	binary
		F	fixed point
		G	floating point
		P	packed decimal
		U	unpacked decimal
X’00’	8	options (continued)	
		bits 1-3	(unused)
		bit 4=1	XI (exclude PE occurrence number from UQ) option
		bits 5-8	(unused)

A hyperdescriptor parent-field name list is an extension of a hyperdescriptor definition. It has the following format for all eight-byte groups after the first:

Notation	Bytes	Explanation
X’00’	1	X’00’ indicates continuation
X’00’	2	(unused)
p-fieldname-list...	3-8	parent-field name list: each name is two bytes; six bytes total (that is, three names. If fewer than three names are provided, the additional bytes are filled with X’00’).

Phonetic Descriptor Definition

’P’ desc-name option p-field-name X’0000’

Notation	Bytes	Explanation
’P’	1	’P’ indicates phonetic descriptor
desc-name	2-3	phonetic descriptor name
option	4	(unused)
p-field-name	5-6	parent-field name
X’0000’	7-8	(unused; set to nulls)

Subfield/Subdescriptor Definition

’S’s-name option p-field-name from to

Notation	Bytes	Usage	
’S’	1	’S’ indicates subdescriptor/subfield	
s-name	2-3	subdescriptor/subfield name	
option	4	definition options:	
		bit 1=1	descriptor
		bit 2=1	exclude PE occurrence number from UQ
		bit 3=1	multiple-value format
		bit 4=1	null-value suppression
		bit 5=1	periodic-group field
		bit 6-7	(unused)
		bit 8=1	unique descriptor
p-field-name	5-6	parent-field name	
from	7	starting (inclusive) byte	
to	8	ending (inclusive) byte	

Superdescriptor/Superfield Definition

```
'T'sup-name option p-field-name from to
X'00' X'000000'p-field-name from to
```

Notation	Bytes	Usage	
'T'	1	'T' indicates superdescriptor/superfield	
sup-name	2-3	superdescriptor name	
option	4	definition options:	
		bit 1=1	descriptor
		bit 2=1	exclude PE occurrence number from UQ
		bit 3=1	multiple-value format
		bit 4=1	null-value suppression
		bit 5=1	periodic-group field
		bit 6-7	(unused)
		bit 8=1	unique descriptor
p-field-name	5-6	parent-field name	
from	7	starting (inclusive) byte	
to	8	ending (inclusive) byte	

Extension of a superdescriptor or superfield definition has the following format on all eight-byte groups after the first:

Notation	Bytes	Explanation
X'00'	1	indicates continuation
X'000000'	2-4	(unused)
p-field-name	5-6	parent-field name
from	7	starting (inclusive) byte
to	8	ending (inclusive) byte

Command Option 2 is Set to "I"

If "I" is set for command option 2, field information is returned in Adabas internal format:

Bytes	Contents
1	X’80’
2	B’00000xyz’ where xyz are ciphering bits: 1=yes; 0=no
	x user y new z old
3-4	total length of FDT, FDT upper index (UI), and SDT (length bytes included)
5-6	FDT length, in bytes (length bytes included)
7-n	FDT field descriptor elements (10 bytes per element; see the following description)
n+1 & n+2	length of FDT upper index (UI) structure (including length bytes)
n+3 to n+3+(n+1/n+2)	upper index (UI)
UI & UI+1	SDT table length (including length bytes)
UI+2 to UI+2 + (UI/UI+1)	SDT table structure

FDT elements beginning in byte 7 each have the format of the FDTE DSECT:

Element Byte	Format
0	level definition
1 - 2	field name
3	field length
4	field format
5	descriptor definition
6	periodic group descriptor
7	security control byte
8 - 9	FDT element pointer

The meaning of FDT elements is described in the Adabas architecture training information.

The FDT is stored in up to four Associator blocks. Therefore, the maximum possible record buffer length is the length of four Associator blocks.

Command Option 2 is Set to Neither S Nor I

Note:
Command option 2 may be set to values other than "I" or "S" to support older programs; these values do not support newer features. Software AG recommends "I" or "S" for new programs.

If the command option 2 field contains neither "I" nor "S", the field information returned *excludes* collation descriptor and sub-/super-/hyper-/phonetic descriptor information. The information is provided in the same format as provided in Adabas version 4:

```
n field-def
```

where:

n	is the number of fields in the file. The number is provided as a four-byte binary number in the first four bytes of the record buffer.
field-def	is the field definition information for each field within the file. The information for each field is provided in six bytes according to the following format:

Bytes	Usage
1	level number (binary)
2 - 3	name (alphanumeric)
4	standard length (binary)
5	standard format (alphanumeric):
	A alphanumeric
	B binary
	F fixed point
	G floating point
	P packed decimal
	U unpacked decimal
	W wide-character
6	definition options:
	bit1=1 descriptor
	bit 2=1 fixed storage
	bit 3=1 multiple-value field
	bit 4=1 null-value suppression
	bit 5=1 periodic group field
	bit 6=1 phonetic source field
	bit 7=1 sub-/superdescriptor source field
	bit 8=1 unique descriptor

The information for the next field immediately follows the information for the preceding field with no intervening spaces.

Example

The field definition information for file 1 is to be read.

Control Block

Command Code	LF	
File Number	1	field definitions requested for file 1
Record Buffer Length	100	
Command Option 2	S	information for all types of descriptors and sub/superfields is to be returned
Additions 3	bbbbbbbb	file is not security-protected

N1 / N2 Command: Add Record

The N1/N2 commands are used to add a new record to a file.

The N1 command adds a new database record with an ISN assigned by Adabas. The N2 command adds a new database record with the ISN assigned by the user.

This chapter covers the following topics:

- Function and Use
- Command: N1 / N2
- Control Block
- Buffers
- Examples

Function and Use

The user specifies the file to which the record is to be added, and the fields for which values are being provided. Any fields not specified will contain a null value in the record added.

Adabas assigns the record an ISN, adds the record to Data Storage, and performs any Associator updating which may be required.

The N2 command is used if the ISN to be assigned to the record is being provided by the user. To keep the ISN assigned to the same record, the unloaded file must be reloaded with the USERISN=YES option.

If the user is an ET logic user and is operating in multiuser mode, the record added is placed in hold status.

Command: N1 / N2

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	A/U *
ISN LOWER LIMIT	17-20	binary	--	A
ISN QUANTITY	21-24	binary	--	A
FORMAT BUFFER LENGTH	25-26	binary	F	U
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-44	--	--	--
ADDITIONS 2	45-48	alphanumeric	--	A
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
ADDITIONS 5	65-72	alphanumeric	F	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	F	U
RECORD BUFFER	F	U

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Filled in by Adabas for N1; unchanged for N2
--	Not used

Control Block

Command Code

N1/N2

Command ID

If a series of records is being added using a series of N1/N2 calls, and the same fields are specified in the format buffer for each call, this field should be set to a non-blank, non-zero value. This results in a reduction in the time required to process each N1/N2 call.

If only a single record is being added, or if the format buffer is modified between N1/N2 calls, this field should be set to blanks.

The first byte of this field may not be set to hexadecimal ’FF’.

File Number

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

If the N1 command is being executed, Adabas returns the ISN assigned to the record in this field.

If the N2 command is being executed, the ISN to be assigned to the record must be provided in this field. The ISN provided must not already be assigned to a record in the file, and must be within the limit (MAXISN) in effect for the file. MAXISN is set by the DBA when the file is loaded.

Note:
You cannot assign an ISN that is greater than the value specified by the MAXISN parameter for the file.

ISN Quantity/Lower Limit

These fields are set to nulls following completion of the N1/N2 command operation.

Format Buffer Length

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified.

Record Buffer Length

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified.

Additions 2: Length of Compressed Record

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the newly added compressed record;
- If the N1 or N2 command returns a nonzero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the additions 5 field is set to 0, the value provided in the command ID field will be used as the format ID as well.

If the high-order bit is set to 1, the contents of the rightmost (low-order) four bytes of the additions 5 field (additions 5 + 4(4)) are interpreted as the format ID.

If the two high-order (leftmost) bits of the first byte of additions 5 field are set to one (B’11’), the value in all eight bytes of the additions 5 field (additions 5 + 0(8)) is used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Buffers

Format Buffer

The fields for which values are being provided in the record buffer must be specified in this buffer. When performing an N1 command, the format buffer cannot contain any of the following:

- format selection criteria ("field-name operator value...");
- an edit mask element;
- a reference to a sub-/superdescriptor field;
- the same field specified more than once (except a multiple-value field);
- an "-N" type of record format specification (for example, ABN or AB1 - N).

Any of the above in the format buffer cause a nucleus response of 44 for an N1 command. Any fields that are not specified in the format buffer will contain a null value in the record being added. The syntax and examples of format buffer construction are provided beginning on page .

The following rules control the processing of non-NU descriptors which are not specified in the format buffer for an N1/N2 command:

- Any omitted non-NU descriptor whose definition in the field definition table (FDT) is both the farthest from the beginning of the FDT, and *precedes* the FDT definition of the last field specified in the format buffer *will* have null values entered in the inverted list for the descriptor;
- Any omitted non-NU descriptor whose definition in the field definition table (FDT) is both the farthest from the beginning of the FDT, and *follows* the FDT definition of the last field specified in the format buffer will *not* have null values entered in the inverted list for the descriptor.

Therefore, the format buffer entry should reference either all non-NU descriptors, or at least one field following (in FDT order) all non-NU descriptor fields. This ensures that null values are correctly inserted in the inverted lists for all non-NU descriptors.

For non-NU descriptors that are contained in a periodic group, null values are entered in their inverted lists only for null occurrences that precede the highest occurrence number specified in the format buffer.

The following additional format buffer considerations are applicable for the N1/N2 command:

1. Subdescriptors, superdescriptors, hyperdescriptors, and phonetic descriptors may not be specified in the format buffer. Adabas automatically creates the correct value for any of the above if a field from which such a descriptor is derived is specified in the format buffer.

2. Theoretically, the maximum record length permitted is 32767 bytes before compression. The actual maximum is limited by block size restrictions. It is also smaller depending on the size of the LU parameter specified for the Adabas session; the maximum is (LU - format buffer length - 108). The maximum record length after compression is equal to the smaller of either the Data Storage block size - 4 bytes, or the Work block size - 110 bytes.

3. If a field is specified using a length override that exceeds the standard length (not permitted if the field is defined with the fixed storage option), all subsequent references to this field should specify the length that was used. If a subsequent reference uses the standard length, value truncation for alphanumeric fields or a non-zero response code for numeric fields may occur.

4. Only a multiple-value field may be specified more than once in the format buffer.

5. A multiple-value count field, periodic group count field, or literal value specified in the format buffer is ignored by Adabas. The corresponding value in the record buffer is also ignored.

6. If a multiple-value field is specified in the format buffer, Adabas sets the multiple-value field count according to the following rules:

   - For a multiple-value field defined with the NU option, the count field is adjusted to reflect the number of existing nonblank values. Blank values are completely suppressed.

Field definition	01,MF,5,A,MU,NU
Format buffer	MF1-3
Record buffer	XXXXXYYYYYZZZZZ
Result after add	XXXXX,YYYYY,ZZZZZ MF count = 3
Format buffer	MF1-3
Record buffer	XXXXXbbbbbZZZZZ
Result after add	XXXXX,ZZZZZ MF count = 2
Format buffer	MF1-3
Record buffer	bbbbbbbbbbbbbbb (blanks)
Result after add	values suppressed MF count = 0

   - For a multiple value field defined without the NU option, the count is adjusted to reflect the number of existing values (including null values).

Field definition	01,MF,5,A,MU
Format buffer	MF1-3
Record buffer	XXXXXYYYYYbbbbb
Result after add	XXXXX,YYYYY,b(blank) MF count = 3
Format buffer	MF1
Record buffer	bbbbb (blanks)
Result after add	b (blank) MF count = 1

Up to 191 values are permitted for a multiple-value field.

7. If a periodic group or a field within a periodic group is specified in the format buffer, Adabas sets the periodic group count equal to the highest occurrence number specified in the format buffer. If the highest occurrence suppresses null values, the count is adjusted accordingly.

Field definitions	01,GB,PE 02,BA,1,B,DE,NU 02,BB,5,P,NU
Format buffer	GB1-2.
Record buffer	X’08000000500F09000000600F’
Result after add	GB (1st occurrence) BA = 8 BB = 500 GB (2nd occurrence) BA = 9 BB = 600 GB count = 2
Format buffer	GB1-2.
Record buffer	X’00000000000F00000000000F’
Result after add	GB (1st occurrence) values suppressed GB (2nd occurrence) values suppressed GB count = 0

Up to 191 occurrences are permitted for a periodic group.

8. 8. If a field defined with variable length (no standard length) is specified in the format buffer, the corresponding value in the record buffer must be preceded by a 1-byte binary number that represents the length of the value (including the length byte).

Field definitions	01,AA,3,A 01,AB,A
Format buffer	AA,AB.
Record buffer	X’F1F2F306F1F2F3F4F5’

Fields AA and AB are to be added. The value for AA is "123". The value for AB (which is a variable length field) is "12345".

Record Buffer

The value for each field specified in the format buffer must be provided in the record buffer.

Each value must be provided according to the standard length and format of the field for which the value is being provided, unless a different length and/or format is specified in the format buffer.

If the field is defined as a variable-length field (no standard length), a 1-byte binary field containing the length of the field (including the length byte) must be provided immediately before the value.

If the field for which the value is being provided is defined as a unique descriptor, the value provided must not already exist for the descriptor; otherwise, the command will be rejected.

Examples

Example 1

A record is to be added to file 1. The ISN of the record is to be assigned by Adabas. The field values which are to be provided are as follows:

Field	Value
AA	ABCD
MF (value 1)	AAA
MF (value 2)	BBB
BA (1st occurrence)	5
BA (2nd occurrence)	6

Control Block

Command Code	N1	
Command ID	bbbb	only 1 record being added
File Number	1	
Format Buffer Length	15	or larger
Record Buffer Length	16	or larger
Additions 3	bbbbbbbb	file 1 is not security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	AA,MF1-2,BA1-2.
Record Buffer	X'C1C2C3C440404040C1C1C1C2C2C20506'

Example 2

A record is to be added to file 2. The ISN of the record is to be provided by the user. The field values to be provided are as follows:

Field	**Value**
RA	12345678
RB	ABCD

Control Block

Command Code	N2	
Command ID	bbbb	only 1 record is to be added
File Number	2	
ISN	20	ISN 20 is to be assigned to the record
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger
Additions 3	password	file 2 is security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	RA,RB.
Record Buffer	X’F1F2F3F4F5F6F7F8C1C2C3C4404040404040’

OP Command: Open User Session

The OP command indicates the beginning of a user session.

This chapter covers the following topics:

- Function and Use
- User Types
- Command: OP
- Control Block
- Record Buffer
- User Queue Element
- Exceeding Time Limits
- Values Returned in Control Block Fields
- Examples

Function and Use

Software AG recommends that all users start their Adabas sessions with an OP command.

An OP command is mandatory if any of the following is true for the user:

- The nucleus is run with ADARUN parameter OPENRQ=YES.
- Exclusive file control (EXF) is to be performed.
- User data that was stored in an Adabas system file by a previous ET command is to be read.
- User data is to be stored in an Adabas system file, using a C3, CL, or ET command.
- The user is to be assigned a special processing priority.
- The user is to be an access-only user (no update commands permitted).
- A transaction time limit and/or a non-activity time limit is to be set for the user that differs from that specified by ADARUN parameters TT and/or TNAx, respectively. The setting for a user must conform to the maximum (2-byte) setting set by the ADARUN parameters MXTT and MXTNA, respectively.
- Special data encoding and/or architecture is to be specified for the user session.

An OP command is otherwise optional. An implicit OP command is issued by Adabas when the first Adabas command is issued by a user who is not currently identified to Adabas.

Users accessing files that are protected by Adabas Security are not required to issue an OP command; such users must, however, provide a password with each command directed to a security-protected file. If the system is protected through the Adabas External Security Interface (ADAESI) and an attached security package, an OP command may be required. Ask your DBA or system security specialist for more information.

If an OP command is issued by an *active* ET logic user, and the user is not at ET status (an OP, ET, or BT command has not been previously issued with one or more records in hold status), Adabas issues a BT command for the user and returns response code 9 for the OP command. If an OP command is issued by any other type of *active* user, Adabas issues a CL command for the user before processing the OP command.

A user operating in single-user mode cannot issue more than one OP command during a given session execution.

User Types

Adabas identifies each user session by the type of access/update performed by the user.

Access-Only Users

If an OP command is issued in which access-only (ACC parameter) is specified, the user is defined to be an access-only user. Such users may not issue hold, update, delete, add record, ET, or BT commands.

Note:
Access-/update-level security can also be controlled through Adabas Security on a file and field level, and through Adabas SAF Security on a database and file level. The security control "adds to" the user type control; that is, an access-only user’s access level can be further defined on a file, field, or value level with Adabas Security, but cannot be changed to an update level.

Exclusive Control Users

If an OP command is issued in which exclusive file control (EXF or EXU parameter) is specified, the user is defined to be an exclusive control user. Such a user is considered to be a non-ET logic user (unless the UPD parameter has also been specified in which case the user is defined to be an ET logic user). If such a user issues an ET command, the user is changed to an ET logic user when the first ET command is issued.

ET Logic Users

All other users (including users who do not issue an OP command) are defined as ET logic users. Transactions issued by such users are subject to the transaction duration time limit.

Command: OP

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	--	A
	9-10	--	--	--
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN LOWER LIMIT	17-20	binary	F	A
ISN QUANTITY	21-24	binary	F	A
	25-26	--	--	--
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-34	--	--	--
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	U
ADDITIONS 2	45-48	alphanumeric	--	A
	49-56	--	--	--
ADDITIONS 4	57-64	binary	F	A
ADDITIONS 5	65-72	binary	--	A
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	*	
RECORD BUFFER	F	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of the call statement
--	Not used

Control Block

Command Code

OP

Command ID

Adabas will return binary zeros in this field if the previous session for this user was terminated successfully with a CL command, or no previous session existed for this user.

If the previous session for this user was not terminated successfully with a CL command, Adabas will return in this field the transaction sequence number of the last successfully completed user transaction.

The above information is returned only if the user is an ET logic user. If the user is a non-ET logic user, this field is not modified by Adabas.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

Adabas sets this field to binary zero on return.

ISN Lower Limit: Non-activity Time Limit

This field may be used to provide a user-specific non-activity time limit. This limit must conform to the maximum specified by the ADARUN parameter MXTNA. If this field contains binary zeros, the non-activity time limit specified by the appropriate ADARUN parameter (TNAA, TNAE, or TNAX) for the Adabas session is in effect.

Following successful OP completion, Adabas returns its system and call type information in this field; the timeout information previously held here is returned in the additions 5 field, bytes 4 and 5. See the section *Values Returned in Control Block Fields* for detailed information.

ISN Quantity: Transaction Time Limit

This field may be used to provide a user-specific transaction time limit. This limit must conform to the maximum specified by the ADARUN parameter MXTT. If this field contains binary zeros, the transaction time limit specified by the ADARUN TT parameter for the Adabas session is in effect.

Following successful OP completion, Adabas returns its system release information in this field; the timeout information previously held here is returned in the additions 5 field, bytes 6 and 7. See the section *Values Returned in Control Block Fields* for detailed information.

Record Buffer Length

The length of the record buffer must be specified in this field. The length specified must be large enough to accommodate all required record buffer entries. The record buffer length should be set to zero if an empty record buffer is to be supplied.

If user data which are stored in an Adabas system file are to be returned, the length specified must be large enough to permit the user data to be inserted in the record buffer; otherwise, the user data will be truncated.

Command Option 1: Restrict Files Option

Option	Description
R	(restrict files) the user is restricted to the files specified in the file list provided in the record buffer. The OP command returns a response code 48 if a specified file is not available. Later commands issue response code 17 if the user attempts to access/update a file not contained in the file list. If the "R" option is not specified and an attempt is made to access a file not currently in the specified file list, Adabas tests to determine whether the file is currently in use by an Adabas utility and if not, the file is added to the user's list. See the section *User Queue Element* for more information.

Command Option 2: Read User Data

Option	Description
E	user (ET) data stored in an Adabas system file by the last successful C3, CL, or ET command issued by the user (in which user data was provided) is returned in the record buffer. This option may only be used if the user has provided the same user ID for this user session and the session during which the user data was stored.

Additions 1: User ID

This field may be used to provide a user ID for the user session. To avoid later limitations, Software AG recommends that you always specify a user ID.

The value provided for the user ID should be unique for the user (that is, not used by any other user at the same time), and must begin with the value "A" through "9". If the value is not unique, a response code 48 occurs. If the other user has been inactive for 60 seconds, the nucleus schedules an internal autobackout for that user and returns response code 9 for the OP command. With another OP command, the user can take over the user ID (ETID) of the other user, who loses the user ID due to the internal backout transaction (BT) command and receives response code 9 on the next call.

A user ID *must* be provided if any of the following is true:

- The user intends to first read (if any) and then store user data, and the user wishes the data to be available during a subsequent user or Adabas session.

  A user that specifies a user ID (ETID) can store ET data that remains available in later sessions. ET data stored by a user having no user ID is available during the user's current session only. Data to be stored can be provided with the ET command or at the end of the session with a close (CL) command. ET data stored with a user ID in a previous session can be read with either the OP or RE command.

- The user is to be assigned a special processing priority (priorities are assigned with Adabas Online System or the ADADBS utility's PRIORITY function);

- The operation being started is on a multiclient file; see the *Adabas DBA Reference* documentation for more information about multiclient files. A user with a blank or missing owner ID receives response code 3 or 113 when trying to access a multiclient file.

Users for whom none of the above conditions are true may set this field to blanks.

Additions 2: Transaction Sequence Number

Adabas returns in this field the transaction sequence number of the last transaction for which an ET command with ET data was executed successfully.

If the OP command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Additions 4: Maximum Settings

This field may be used to set the following user-specific maximum binary values:

Bytes	Usage
57-58	maximum number of ISNs that may be stored in the internal ISN element table resulting from the execution of a Sx command. Increasing the default setting will result in less access to the Adabas Work being necessary. The maximum allowed is 1000.
59-60	maximum number of records that a user may have in hold status at the same time. The default is the value set by the ADARUN NISNHQ parameter, and the maximum is 1/4 the value set by the ADARUN NH parameter minus 1, or 65535-whichever is smaller.
61-62	maximum number of command IDs that may be active for a user at the same time. This value cannot be greater than 1/240 &#65533; LQ (where LQ is the ADARUN sequential command table length parameter value, which has a default of 10000).
63-64	maximum amount of time permitted for the execution of an Sx command.

If one or more of the values above are not specified, the system-wide specifications for the nucleus become the defaults. The user should consult with the DBA concerning the system defaults in effect for these values before entering any user-specific values in this field.

Values specified must be in binary. Specifying blanks or binary zero is equivalent to "no value".

Adabas sets the additions 4 field to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Returned Time-Out Values

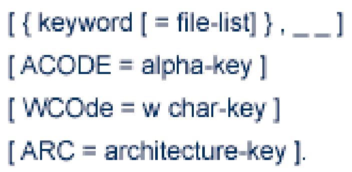
User-specific timeout values are returned in the left (high-order) and right (low-order) halves, respectively, of the rightmost fullword of the additions 5 field. For more information, see the section *Values Returned in Control Block Fields*.

Record Buffer

The record buffer specifies

- the files to be accessed and/or updated, and the type of updating to be performed
- special encoding for alphanumeric and/or wide-character fields during the session
- for fields in record and value buffers, special architecture that overrides the architecture for remote calls set by Entire Net-work.

The syntax of the record buffer is



```
[ { keyword [ = file-list] } , _ _ ]
[ ACODE = alpha-key ]
[ WCOde = w char-key ]
[ ARC = architecture-key ].
```

Record Buffer Syntax

-where:

- "keyword" is one of the following:

ACC ESS	file is to be accessed only
UPD ATE	file is to be updated (implies ET logic)
EXF	exclusive file control: no other users may access/update the file
EXU	file is to be updated under exclusive control of the user. No other user can update the specified file while this user session is active. Exclusive control is given only if no other active user has issued an OP command specifying the EXF/EXU or UPD parameter for the file.

- "file-list" is one or more 1-5-digit file numbers (leading zeros permitted) indicating the Adabas file(s) for which the preceding file-list keyword is applicable.
- ACODE assigns special encoding for A fields during the user session.
- "alpha-key" specifies the key of supplied encoding descriptor objects.
- WCODE assigns special encoding for W fields during the user session.
- "wchar-key" specifies the key of supplied encoding descriptor objects.
- ARC defines special data architecture for fields in the record and value buffers. This definition overrides the architecture key defined for remote calls in Entire Net-work.
-

  Note:
  The ARC setting does not affect the data coversion performed by ADALNK / LNKUES on the Adabas control block (ACB) and buffers with fixed layout such as the search and format buffers.

- "architecture-key" is an integer which is the sum of the following numbers:

byte order	b=0	high-order byte first
	b=1	low-order byte first
encoding family	e=0	ASCII encoding family
	e=2	EBCDIC encoding family (default for local calls)
floating-point format	f=0	IBM370 floating-point format
	f=4	VAX floating-point format
	f=8	IEEE floating-point format

The default is ARC = b + e + f = 2; that is, high-order byte first; EBCDIC encoding family; and IBM370 floating-point format (b=0; e=2; f=0).

User data from an Intel386 PC provides the example: b=1; e=0; f=8; or ARC=9.

The record buffer syntax must end with "."

If no parameters are specified for the record buffer, it contains only "." In this case, the record buffer length can be set to zero in the Adabas control block and the record buffer need not be supplied.

If a user-type "keyword" is to apply to a series of files, each file for which the keyword is applicable may be specified with a comma between each file number. Duplicate file numbers within a keyword are permitted. Duplicate file numbers across keywords are permitted. Each keyword may appear only once.

UPD and EXU also imply access to the file. If ACC is the only keyword specified, a file list is not required.

Note:
For Natural Lightstorm (NLS) version 3.1 users: it is necessary to enter a file list in the record buffer of the OP command because NLS sets option ’R’ (restricted open) leading Adabas to expect a file there.

If no user-type keyword is specified, the user automatically becomes an ET logic user.

User Queue Element

During the time that a user is active, Adabas maintains a user queue element (UQE) for the user.

User Type and File Lists

The UQE lists the numbers of up to 5000 files the user is currently using. For a non-ET user, Adabas creates the file list when the user issues an OP command; no file list is created for an ET user. The file list may be modified during the user session. If no OP command is issued, the file list will initially contain no files. Each file in the file list is marked with one of the following use classes:

- ACC, access only;
- EXF/EXU, access and update and under exclusive control;

- UPD, access and update;
- UTI, access and update and in use by an Adabas utility.

If a subsequent attempt is made to access a file not currently in the specified file list, Adabas tests to determine whether the file is currently in use by an Adabas utility. If not, the file is added to the user's UQE and marked as ACC. However, if the OP command specifies the restrict files option in the command option 1 field with a file list in the record buffer, and then tries to access a file *not* in the file list, response code 17 occurs.

If a subsequent attempt is made to update a file not currently in the user's file list, the following tests are applied:

- Does the request conflict with the user type? For example, an access-only user may not issue update commands.
- Is the file to be updated under exclusive control of another user or Adabas utility?

If the file is determined to be available for the user, the file is added to the user's UQE and marked as UPD.

Special Encoding Information

If the user specifies ACODE, WCODE, and/or ARC to determine the special encoding to be used, this information is communicated to the Adabas nucleus, which stores it in the UQE.

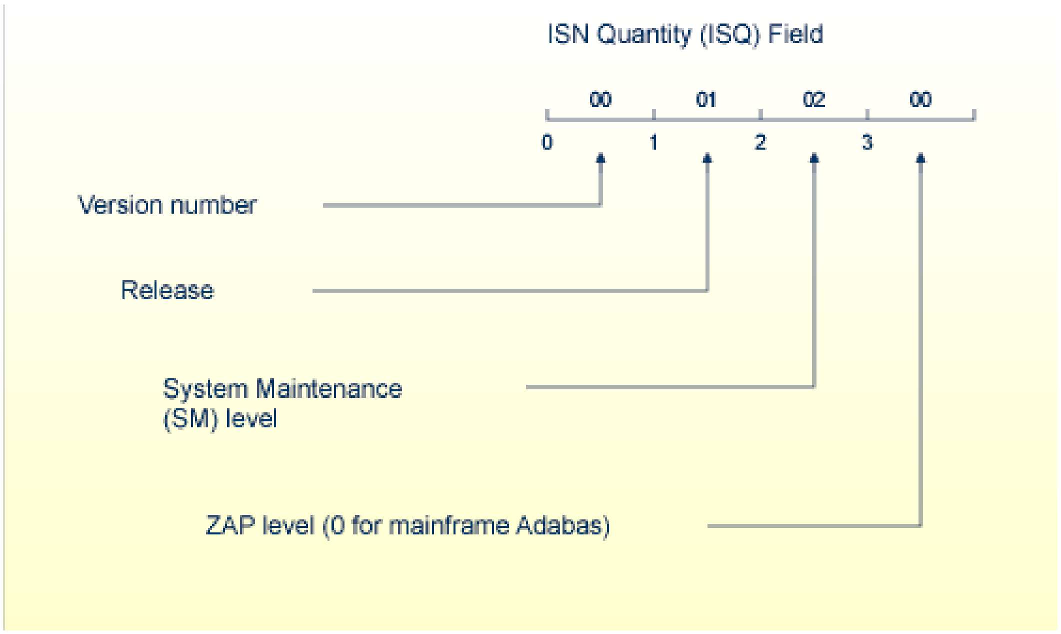
Exceeding Time Limits

See the section on timeout characteristics in the *Adabas Operations* documentation for information on which action will be taken if a user exceeds the non-activity time limit.

Values Returned in Control Block Fields

In some cases, values are returned in the ISN lower limit and ISN quantity fields to allow compatibility with VMS/UNIX systems. As a result, any user-specific timeout values previously held in these fields in early Adabas releases are now returned in the additions 5 field. These changes also affect the corresponding command log fields.

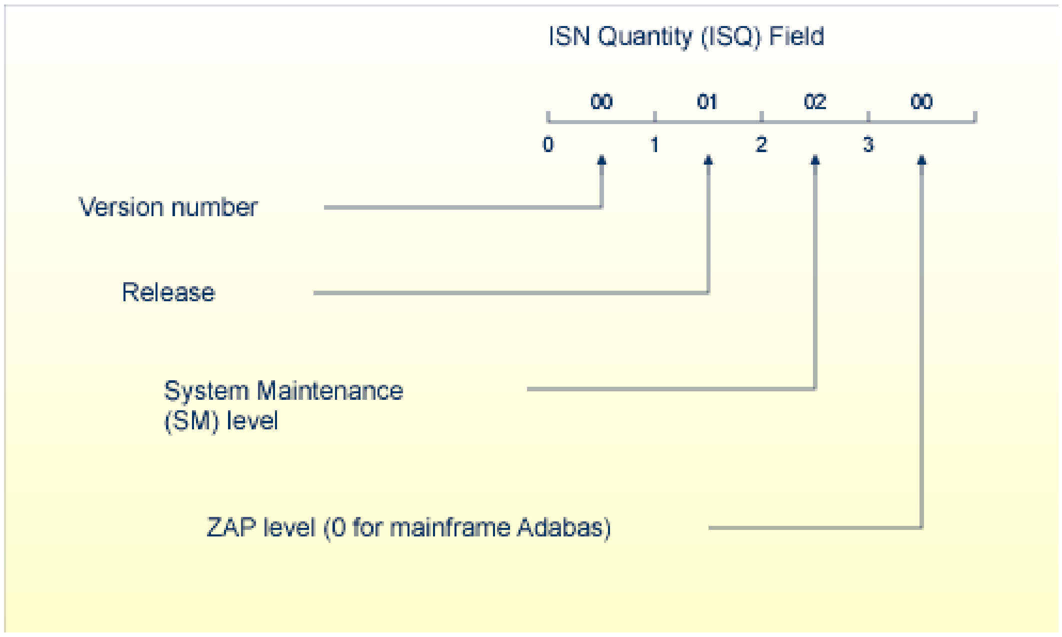
The ISN quantity field returns the following binary values after OP execution:


ISN Quantity (ISQ) Field
00
01
02
00
0
1
2
3
Version number
Release
System Maintenance (SM) level
ZAP level (0 for mainframe Adabas)

Binary Values Returned after OP Execution

This returned information provides compatibility with VMS/UNIX.

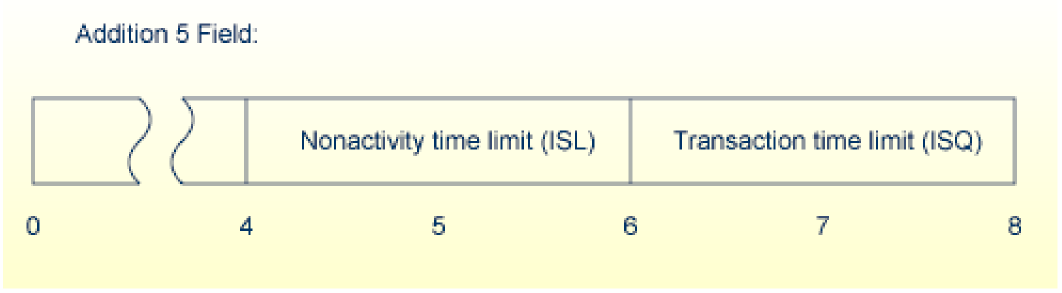
The ISN lower limit (ISL) field now returns the following binary information:


ISN Quantity (ISQ) Field
00
01
02
00
0
1
2
3
Version number
Release
System Maintenance (SM) level
ZAP level (0 for mainframe Adabas)

Binary Information Returned from ISL

The rightmost two bytes indicate whether the user program is running in a noncluster (00) or cluster (nn) nucleus environment. The specific cluster nucleus type return code is for use by Software AG technical support.

Any timeout values specified in the ISL/ISQ fields at the start of OP processing are returned in the additions 5 field, as follows:



Timeout Values

and correspondingly, in the additions 5 command log entry.

Examples

Example 1: Access-Only User

An access-only user session is to be opened.

Control Block

Command Code	OP	
Record Buffer Length	4	or larger

Buffer Areas

Record Buffer	ACC. (or ACC= . or ACC=file-list.)

Allows all selected files to be accessed.

Example 2: ET Logic User

A user session is to be opened during which the user intends to access files 8 and 9 and update files 8 and 16. The user intends to store user data in an Adabas system file during the session. The user data stored during the previous session are to be read. The ID for the user is "USER0001".

Control Block

Command Code	OP	
Record Buffer Length	15	or larger
Command Option 2	E	user data are to be read
Additions 1	USER0001	user ID is required if user data are to be stored and/or read

Buffer Areas

Record Buffer	ACC=9,UPD=8,16.

Example 3: Exclusive Control User Without ET Logic

A user session is to be opened during which the user wishes to have exclusive control of files 10, 11, and 12. The user does not intend to use ET commands and does not intend to store and/or read user data in/from an Adabas system file.

Control Block

Command Code	OP	
Record Buffer Length	13	or larger
Command Option 2	b	user data are not to be stored or read
Additions 1	bbbbbbbb	user ID is not required

Buffer Areas

Record Buffer	EXU=10,11,12.

Example 4: Exclusive Control User with ET Logic

A user session is to be opened during which the user wishes to have exclusive control of files 10,11, and 12. The user intends to use ET commands.

Control Block

Command Code	OP	
Record Buffer Length	26	or larger
Command Option 2	b	user data are not to be stored or read
Additions 1	bbbbbbbb	user ID is not required

Buffer Areas

Record Buffer	EXU=10,11,12,UPD=10,11,12.

Example 5: Special Encoding for Wide-Character Fields

A user session is to be opened with shift-JIS special encoding for wide-character fields. The user intends to update file number 1.

Control Block

Command Code	OP	
Record Buffer Length	16	or larger

Buffer Areas

Record Buffer	UPD=1,WCODE=932.

RC Command: Release Command ID or Global Format ID

The RC command releases one or more command IDs or a global format ID for the issuing user.

This chapter covers the following topics:

- Function and Use
- Command: RC
- Control Block
- Examples

Function and Use

The RC command may be used to release one or more command IDs currently assigned to a user, or to delete one or all global format IDs, as follows:

- internal format buffer pool command IDs. Related internal formats are also released;
- ISN list (TBI) command IDs;
- command IDs in the table of sequential commands (TBLES/TBQ);
- command IDs equal to and greater than the specified command ID value in either the internal format buffer pool or the TBI, TBLES and TBQ, or both;
- one "special" global format ID for a user group;
- all existing global format IDs.

If no selective options are specified, the entered command ID is released from all of the above areas. When a command ID is released, its related TBI and/or TBLES/TBQ entries are also removed; however, the internal format buffer pool entry is not necessarily released.

The RC command should be used under the following conditions:

- The user has completed processing an ISN list stored on the Adabas Work by an Sx command that specified the save-ISN-list option. Issuing the RC command permits Adabas to reuse the space currently occupied by the list;
- The user wishes to terminate a sequential pass of a file (using an L2/L5, L3/L6, or L9 command) before reaching an end-of-file condition;
- The user has completed a series of L1/L4, A1/A4, or N1/N2 commands in which a non-blank command ID was used.

Command: RC

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
	13-34	--	--	--
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	U
	45-64	--	--	--
ADDITIONS 5	65-72	alphanumeric	F	U
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Not used

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block

Command Code

RC

Command ID

The command ID to be released or to be used as a reference is specified in this field. A value of blanks or binary zeros releases all the command IDs currently assigned to the user.

File Number

If command option D, E, or O is specified, the file number field must contain the binary number of the file associated with the format or global format ID to be released.

For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

Command Option 1/2: Type of Command IDs to Be Released

These fields are used to indicate that a command ID, format ID, or global format ID is to be released from the internal format buffer pool, the ISN list table (TBI), or the table of sequential commands (TBLES/TBQ). For information about the tables, see the section *General Programming Considerations*.

If both command option 1/2 fields are set to blanks or binary zeros, the command ID specified in the command ID field is released from all tables in which it is present.

If either command option field is set to one of the following values, the resources associated with the command ID, format ID, or global format ID are released as indicated:

Option	Releases . . .
C	all existing global formats
D	all formats for a given file number and descriptor name
E	all global formats for a given file number and descriptor name
F	the format associated with the specified command ID
G	all existing formats associated with command IDs greater than or equal to the specified command ID
I	the ISN list (TBI) associated with the specified command ID
L	the global format ID contained in the additions 5 field.
O	the global format ID contained in the additions 5 field for a given file number.
S	the sequential commands (TBLES/TBQ) associated with the specified command ID
X	all ISN lists (TBI) and sequential commands (TBLES/TBQ) associated with command IDs that are greater than or equal to the specified command ID. Internal formats are not released.

Options D and E are used when the format was created by an L3/L6 command to ensure the return of correct data in an environment where Smith/Jones problems are possible. The underlying format identifier in these cases is 12 bytes: an 8-byte format ID, a 2-byte file number, and a 2-byte descriptor name.

Additions 1: Descriptor Name

If command option D or E is specified, the first two bytes of the additions 1 field must contain the alphanumeric descriptor field name associated with the format or global format ID to be released. All remaining positions must be set to blanks.

If the format to be released was not created using the L3/L6 command, this field is not used.

Additions 5: Released Global Format ID

In this field, specify a global format ID to be released.

Examples

Example 1

The command ID "0003" is to be released.

Control Block

Command Code	RC	
Command ID	X’0003’	command ID 003 to be released
Command Option 1/2	bb	all CID types to be released

Example 2

All command IDs currently assigned to the user are to be released.

Control Block

Command Code	RC	
Command ID	X’00000000’	binary zeros indicate that all command IDs are to be released
Command Option 1/2	bb	all CID types to be released

Example 3

All the command IDs assigned to the user and contained in the table of sequential commands or the internal format buffer pool are to be released.

Control Block

Command Code	RC	
Command ID	X’00000000’	binary zeros indicate that all command IDs are to be released
Command Option 1	F	F indicates that command IDs contained in the internal format buffer pool are to be released
Command Option 2	S	S indicates that command IDs contained in the table of sequential commands are to be released

Example 4

The same global format ID is defined for several files. Release it for all files.

Control Block

Command Code	RC	
Command Option 1/2	L	releases the formats of the global format ID contained in the additions 5 field.
Additions 5	C’TGLOB001’	B’11’ in the two high-order (leftmost) bits of the first byte of this number identify all eight bytes as the global format ID.

Example 5

The same global format ID is defined for several files. Release it for the file 3 only.

Control Block

Command Code	RC	
File Number	03	binary number of the file for which the global format ID is to be released.
Command Option 1/2	O	releases the formats of the global format ID contained in the additions 5 field for the file specified in the file number field.
Additions 5	C’TGLOB001’	B’11’ in the two high-order (leftmost) bits of the first byte of this number identify all eight bytes as the global format ID.

RE Command: Read ETuser Data

The RE command reads ET (user) data for the current user, another user, or all users.

This chapter covers the following topics:

- Function and Use
- Command: RE
- Control Block
- Record Buffer
- Examples

Function and Use

The RE command reads user data that was previously stored in an Adabas system checkpoint file by a C3, CL, or ET command. The user data is returned in the record buffer. These user data may be needed for a user restart following abnormal termination of a user or Adabas session.

User data is read for the issuing user if no value is specified in the command option 1 field. The RE command reads user data stored during a previous session for the issuing user if the previous and current sessions both began with OP commands specifying the user ID.

Depending on the specified command option, the RE command reads user data for either another specific user (if that user ID is specified) or for all users.

- If "I" is specified in the command option 1 field, user data stored by another user may be read if the ID of the user who stored the data is specified in the additions 1 field.
- If "A" is specified in the command option 1 field, the current and the following RE commands read all user data for all user IDs in ascending logical sequence. The corresponding user ID is returned in the additions 1 field as each RE command is completed. The RE command with option "A" reads only the user data written to the checkpoint file whose transactions ended with an ET command; user data are *not* read for users’ transactions that are not yet closed with an ET command.

Command: RE

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	binary	--	A
	9-10	--	--	--
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	A
	17-26	--	--	--
RECORD BUFFER LENGTH	27-28	binary	F	U
	29-34	--	--	--
COMMAND OPTION 1	35	alphanumeric	F	U
	36	--	--	--
ADDITIONS 1	37-44	alphanumeric	F *	U/A **
ADDITIONS 2	45-48	alphanumeric	--	A
	49-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	***	--
RECORD BUFFER	--	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Supplied ET data user ID when command option 1 equals 1
**	User ID for ET data in record buffer if command option 1 equals A
***	Not used but must be included in parameter list of call statement
--	Not used

Control Block

Command Code

RE

Command ID

Adabas will return a transaction sequence number or binary zeros in this field.

Adabas returns binary zeros in this field if the user whose user data is to be read is not active *and* either no previous session exists for this user or the previous session for this user was terminated successfully with a CL command. Non-ET-logic users receive binary zeros in this field.

If the user is currently active *or* the previous session for the user was not terminated successfully with a CL command, Adabas returns the transaction sequence number of the last successfully completed user transaction.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. If command option 1 specified "A", a response code 3 indicates end-of-file for the user data. Nonzero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

If command option 1 specifies "A", user data is returned in logical sequence starting with this ISN. If this field specifies zero, all user data is returned.

Record Buffer Length

The length of the record buffer. The length specified determines the number of bytes of user data to be returned.

If the length specified is less than the number of bytes of user data available, only the specified number of bytes are inserted in the record buffer and the rightmost bytes are truncated.

Command Option 1

If no value is specified in the command option 1 field, user data is read for the issuing user. The RE command reads user data stored during a previous session for the issuing user if the previous and current sessions both began with OP commands specifying the user ID.

If a command option 1 value is specified, the RE command reads user data for either another specific user or for all users as follows:

Option	Description
I	(ID of user) reads user data stored by another user if the ID of the user who stored the data is specified in the additions 1 field.
A	(all users)The current and following RE commands read all user data in the record buffer for all user IDs in ascending logical (ISN) sequence. A starting ISN can be specified in the ISN field. If the ISN field specifies zero, all user data is returned. The user ID for the user data contained in the record buffer at the end of the current and each following RE operation is returned in the additions 1 field as each RE command is completed. RE reads only the user data written to the checkpoint file for users whose transactions ended with an ET command; user data are *not* read for users' transactions that are not yet closed with an ET command. For this option, a response code 3 returned in the response code field indicates end-of-file for the user data.

Additions 1: User ID

If user data stored by another user are to be read, this field must be set to the ID of the user who stored the data. If command option 1 specifies "A", this field returns the user ID for the user data contained in the record buffer at the end of this and each following RE operation.

Additions 2: Transaction Sequence Number

If an ET logic user stored the data being read, Adabas will return in this field the transaction sequence number of the user's last successfully completed transaction in which user data were stored with an ET or CL command.

If the RE command returns a non-zero response code, the rightmost two bytes of the additions 2 field may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Record Buffer

Adabas returns the ET user data in the record buffer. If no user data was found, this contains blanks at the end of RE operation.

Examples

Example 1

The user wishes to read the user's own data previously stored with an ET command.

Control Block

Command Code	RE	
Record Buffer Length	100	100 bytes of user data are to be read
Command Option 1	blank	the user data to be read were stored by this user
ISN	0	

Example 2

The user wishes to read user data stored by another user (user ID = USER0002).

Control Block

Command Code	RE	
Record Buffer Length	150	150 bytes of user data are to be read
Command Option 1	I	the user data to be read were stored by another user
Additions 1	USER0002	ID of the user who stored the user data
ISN	0	

Example 3

In the following example, the user wishes to read all user data and the corresponding user ID.

Control Block

Command Code	RE	
Record Buffer Length	250	250 bytes of user data are to be read, per user
Command Option 1	A	
ISN	0	read all user data

RI Command: Release Record

The RI command releases a held record and ISN.

This chapter covers the following topics:

- Function and Use
- Command: RI
- Control Block
- Examples

Function and Use

The RI command releases ISNs for records being held by the issuing user. The selected ISN for a single database file, or all ISNs held by the issuing user in all files can be released.

The user specifies the file and ISN of the record to be released in the appropriate Adabas control block fields. Specifying zeros in the ISN field releases all the records currently being held by the user in all files.

Note:
Programs using ET logic should not release records with the RI command if any updating has been performed during the current transaction, since this could result in a loss of data integrity. ET users should release ISNs with the ET or CL commands.

Command: RI

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
	5-8	--	--	--
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	U
	17-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Not used

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block

Command Code

RI

File Number

The number of the file containing the record to be released.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

The ISN of the record to be released. If an ISN value is entered, the file number must also be specified. To release all ISNs held by the user, set this field to binary zeros.

Examples

Example 1

The record identified by ISN 3 in file 2 is to be released from hold status.

Control Block

Command Code	RI	
File Number	2	record to be released is in file 2
ISN	3	record with ISN 3 is to be released

Example 2

Any records being held by the issuing user are to be released from hold status.

Control Block

Command Code	RI	
File Number	-	a value in this field is ignored if the ISN field is zero
ISN	0	release ISNs for all held records from all files

S1 / S2 / S4 Command: Find Records

The S1 / S2 / S4 commands return a count of records and ISNs of records satisfying the search criterion.

This chapter covers the following topics:

- Function and Use
- Command: S1 / S2 / S4
- Control Block
- Buffers
- Examples

Function and Use

The S1/S2/S4 commands are used to select records that satisfy a given search criterion. They can be performed on Adabas expanded files as well.

The result of an S1/S2/S4 command is the number of records which satisfy the query and a list of the records' ISNs. The S1/S4 commands return the ISNs, sorted in ascending sequence, in the ISN buffer; the S2 command returns the ISNs according to a sort sequence specified in the additions 1 field.

Note:
ISN lists that are not in ascending sequence cannot be processed by a later S8 command.

The following types of searches are possible:

- Single file search. The search criterion consists of one or more fields contained in a single file.
- Multiple file search using physically coupled files. The search criterion consists of fields contained in two or more files that have been physically coupled using the ADAINV utility.
- Search using the soft coupling feature. This feature provides for a combination of search, read, and internal list matching.

A search criterion may contain one or more fields that are not defined as descriptors. If nondescriptors are used, Adabas performs read operations to determine which records to return to the user. If only descriptors are used within the search criterion, Adabas resolves the query by using the Associator inverted lists; no read operation is required.

If a valid command ID is specified, Adabas will store on Adabas Work any ISNs that could not be inserted in the ISN buffer on the initial S1/S2/S4 command. These overflow ISNs may be retrieved to the ISN buffer later with additional S1/S2/S4 commands that specify the same command ID.

Adabas releases an overflow ISN list when the last ISN in the list has been returned to the user. If the user needs to retain the entire ISN list indefinitely, the save-ISN-list option may be used. If this option is specified, the entire ISN list is stored on Adabas Work. The ISN list is not released until either an RC, CL, or Sx command with the release CID option is issued, or the Adabas session is terminated.

If the user intends to use the GET NEXT option of an L1 or L4 command to read the records identified by the ISNs, neither the ISN buffer entry nor the save-ISN-list option is required. In this case, the L1/L4 commands obtain the ISNs automatically from the ISN list stored by Adabas.

By placing field names in the format buffer, the user can read the field’s contents from the record of the first ISN in the resulting ISN list. The field’s contents are read into the record buffer. The S4 command also places the first ISN of the resulting ISN list in hold status.

Command: S1 / S2 / S4

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN LOWER LIMIT	17-20	binary	F	U
ISN QUANTITY	21-24	binary	F *	A
FORMAT BUFFER LENGTH	25-26	binary	F	U
RECORD BUFFER LENGTH	27-28	binary	F	U
SEARCH BUFFER LENGTH	29-30	binary	F	U
VALUE BUFFER LENGTH	31-32	binary	F	U
ISN BUFFER LENGTH	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	U
ADDITIONS 2	45-48	binary / binary	--	A
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
ADDITIONS 5	65-72	alphanumeric	F	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	F	U
RECORD BUFFER	--	A
SEARCH BUFFER	F	U
VALUE BUFFER	F	U
ISN BUFFER	--	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Optional timeout value, in seconds
--	Not used

Control Block

Command Code

S1/S2/S4

Command ID

This value identifies the resulting complete or overflow ISN list stored on Adabas Work, and identifies the format buffer for subsequent commands if the read-first-record option is in effect (see the section *Format Buffer*).

If the save-ISN-list option is to be used, or if overflow ISNs are to be stored, a non-blank, non-zero value must be provided in this field.

The first byte of this field may not be set to hexadecimal ’FF’.

See the section *ISN List Processing* for more information.

File Number

File number specifies the number of the file from which the ISNs are to be selected.

If a query using coupled files is to be performed, the file specified in this field will be considered the "primary" file. The file number of physically coupled files must be no greater than 255.

The search can also be performed on Adabas expanded files.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

Adabas returns the first ISN of the resulting ISN list in this field. If there were no resulting ISNs, this field is set to zeros.

ISN Lower Limit

Use this field in the first Sx call to specify a minimum ISN value for the resulting ISN list. The list will then contain only the ISNs greater than the ISN specified in this field. If this field is set to zeros, Adabas will return all qualifying ISNs.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from Adabas Work.

For sorted ISN lists (see S2 commands), this field must be set either to zero or to a valid ISN.

ISN Quantity

This field defines the maximum number of seconds that can be used for Sx command execution.

As a result of an initial Sx call, Adabas returns the number of records that satisfy the search criterion in this field. If security-by-value is being used, response code 1 is returned in this field along with the value 0 (one record found) or 1 (more than one record found). For more information, see the *Adabas Security Manual*.

As a result of a subsequent Sx call used to retrieve ISNs from Adabas Work, Adabas provides the number of returned ISNs in this field. The ISNs themselves are returned in the ISN buffer.

Format Buffer Length

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified.

Record Buffer Length

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified.

Search Buffer Length

The search buffer length (in bytes). The search buffer area defined in the user program must be as large as (or larger than) the length specified.

Value Buffer Length

The value buffer length (in bytes). The value buffer area defined in the user program must be as large as (or larger than) the length specified.

ISN Buffer Length

The ISN buffer length (in bytes). This length is used to determine the number of ISNs placed in the ISN buffer.

If this field is set to zeros, no ISNs will be inserted in the ISN buffer. This field should be set to zeros if the resulting ISN list is to be read with the GET NEXT option of the L1/L4 command, or if the command is being issued only to determine the number of qualifying records.

If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas will reduce the length to the next lower integer which is a multiple of 4.

Command Option 1

Option	Description
H	(save ISN list) stores the entire ISN list resulting from an Sx command on Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.
R	(return) for an S4 command, returns response code 145 if a record to be read and held is not available.

Command Option 2

Option	Description
D	(descending sequence) for an S2 command, sorts descriptor values in descending sequence.

If no command option 2 is specified for an S2 command, the descriptor values are sorted in ascending sequence.

Command Option 1/2: Release CID Option

The "I" option may be specified in either the command option 1 or command option 2 field:

Option	Description
I	releases the command ID (CID) value specified in the command ID field as the first action taken during command execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Additions 1: S2 Command, Descriptors Used for Sort Control

If the S2 command is being used, this field must specify the descriptor (or descriptors) to be used to control the sort sequence; if no sort argument is specified, the S2 command returns response code 28.

One to three descriptors, including subdescriptors and superdescriptors, can be specified. Phonetic descriptors or descriptors contained within a periodic group cannot be specified. A multiple-value field can be specified, in which case the ISNs will be sorted according to the lowest value present within a given record.

Any unused positions of this field must be set to blanks.

Example

```
XXYYbbbb
```

where:

XX	is the major sort descriptor; and
YY	is the minor sort descriptor

The number of ISNs that can be sorted depends on the size of the sort work area (ADARUN LS parameter) defined by the DBA. If the sort area is too small, no sort will be performed; response code 1 will be returned, and the ISNs will be returned in ascending sequence.

Additions 2: Length of Compressed and Decompressed Record

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

If the Sx command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes* documentation.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code and Version/Nucleus ID

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and the database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID

This field may be used to provide a separate format ID to identify the internal format buffer to be used for this command, or to provide a global format ID.

If the high-order bit of the additions 5 field is zero (0), the value provided in the command ID field is also used as the format ID.

If the leftmost (high-order) additions 5 field bit is one (1), the fifth through last bytes of the additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of additions 5 field are set to one (B’11’), the value in all eight bytes of the additions 5 field (additions 5 + 0(8)) is used as a global format ID (that is, the format ID can be used by several users at the same time).

For more information, refer to the section *Command ID, Format ID, Global Format ID*.

Buffers

Format Buffer

If the record identified by the first ISN in the resulting ISN list is to be read from Data Storage, the fields within the record for which values are to be returned must be specified in this buffer. The syntax and examples of format buffer construction are provided in the section *Adabas Calling Procedure*.
User-specified fields for controlling the S2 command’s ISN sort sequence must be specified in the additions 1 field.

If no read is to be performed, the first non-blank character of this buffer must be a period (.).

If a valid command ID is specified, Adabas retains this decoded format buffer for use by later commands specifying the same command ID.

Record Buffer

If the format buffer contains field definitions to enable the read option, Adabas returns the requested field values in this buffer.

The values are returned according to the standard length and format of the field, unless the user specifies a different length and/or format in the format buffer.

Search and Value Buffers

Search and value buffers are used to define the search criteria. The search expression (or expressions) is provided in the search buffer, and the values which correspond to the search expressions are provided in the value buffer.

The syntax and examples of search and value buffer construction are described in the *Adabas Calling Procedure*.

ISN Buffer

Adabas places the list of resulting ISNs in this buffer. Each ISN is returned as a four-byte binary number. The ISNs are returned in ascending ISN sequence unless the S2 command is being used, in which case they are returned in the user-specified sort sequence.

If the query contains one or more file-coupling criteria, the resulting ISN list contains only those ISNs in the primary file (the file specified in the control block’s file number field).

If the ISN buffer length field is set to less than 4, no ISNs are returned in the ISN buffer. If a valid command ID is specified, the ISN buffer length is not zero, but the ISN buffer is still too small to contain all the resulting ISNs, Adabas will store the overflow ISNs on Adabas Work. These ISNs may then be retrieved using further S1/S2/S4 calls in which the same command ID is used.

See the section *ISN List Processing* for additional information.

Examples

For the Adabas file definitions used in all the examples in this section, see *File Definitions Used in Examples*.

Example 1

Select the records in file 1 that contain a value in the range "A" to "J" for the descriptor AA.

Control Block

Command Code	S1	
Command ID	bbbb	no ISNs are to be stored on the Adabas Work
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be returned
Format Buffer Length	1	or larger
Search Buffer Length	12	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	200	no more than 50 ISNs are expected
Command Option 1	b	save-ISN-list option not used
Additions 3	bbbbbbbb	the file is not security-protected

Buffer Areas

Format Buffer	.	no read to be done
Search Buffer	AA,1,S,AA,1.	
Value Buffer	C’AJ’	

Example 2

Find with read option. Select the ISN of the record containing the value "ABCDEFGH" for the field AA in file 1. Also, read the record from Data Storage and return the value for the field AC.

Control Block

Command Code	S1	
Command ID	bbbb	no ISNs are to be stored on the Adabas Work
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be returned
Format Buffer Length	3	or larger
Record Buffer Length	20	or larger
Search Buffer Length	3	or larger
Value Buffer Length	8	or larger
ISN Buffer Length	4	no more than one ISN is expected
Command Option 1	b	save-ISN-list option not used
Additions 3	bbbbbbbb	file is not security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	AC.	the value for field AC is to be returned
Search Buffer	AA.	
Value Buffer	C’ABCDEFGH’ X’C1C2C3C4C5C6C7C8’	

Example 3

Find with ISN buffer overflow. Select the records that contain any value in the range "A" to "D" for field AA in file 1. Use ISN buffer overflow handling.

Control Block

Command Code	S1	
Command ID	ABCD	a non-blank command ID is required
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be returned
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	12	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	100	up to 25 ISNs will be returned with each call
Command Option 1	b	save-ISN-list option not used
Additions 3	bbbbbbbb	file is not security-protected

Buffer Areas

Format Buffer	.	no read to be done
Search Buffer	AA,1,S,AA,1.	
Value Buffer	C’AD’ X’C1C4’	

Adabas will return, as a result of the initial S1 call, a maximum of 25 ISNs in the ISN buffer. If more than 25 ISNs resulted from the query, the remaining ISNs will be stored on Adabas Work under the command ID "ABCD". These overflow ISNs may be retrieved by repeating the S1 call using the same command ID.

Example 4

Find with save-ISN-list option. Select all the records containing the value "+80" for the field XB in file 2. Store the entire resulting ISN list on the Adabas Work.

Control Block

Command Code	S1	
Command ID	BCDE	a non-blank command ID is required when using the save-ISN-list option
File Number	2	
ISN Lower Limit	0	all qualifying ISNs are to be selected
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	3	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	200	a maximum of 50 ISNs will be returned on each call
Command Option 1	H	save-ISN-list option is to be used
Additions 3	password	file is security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	XB.	
Value Buffer	X’080C’	

The user may retrieve any group of ISNs from the ISN list that is stored as a result of this call by repeating the S1 command using the command ID "BCDE". Adabas will insert as many ISNs as can be accommodated in the ISN buffer starting with the first ISN that is greater than the ISN specified in the ISN lower limit field.

Example 5

Find with sort. Select all records containing a value in the range "A" to "F" for the field AA in file 1. Return the resulting ISN list in ascending order of the values for field AB.

Control Block

Command Code	S2	
Command ID	CDEF	a non-blank command ID is required when using the S2 command
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be selected
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	12	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	100	a maximum of 25 ISNs will be returned on each call
Command Option 1	b	the save-ISN-list option is not used
Command Option 2	b	the descending sort option is not used
Additions 1	ABbbbbbb	resulting ISNs are to be sorted on the values of field AB
Additions 3	bbbbbbbb	file is not security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	AA,1,S,AA,1.	
Value Buffer	C'AF' X'C1C6'	

Example 6

Find with hold. Select the record in file 1 containing the value "87654321" for field AA. Also, read the record and place it in hold status. Return the values for fields AB and AC.

Control Block

Command Code	S4	
Command ID	bbbb	blank command ID may be used since save-ISN-list option is not to be used and no overflow ISNs are expected
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be selected
Format Buffer Length	6	or larger
Record Buffer Length	22	or larger
Search Buffer Length	3	or larger
Value Buffer Length	8	or larger
ISN Buffer Length	4	only one ISN is expected
Command Option 1	b	the save-ISN-list option is not to be used
Additions 3	bbbbbbbb	file is not security-protected
Additions 4	bbbbbbbb	file is not ciphered

Buffer Areas

Format Buffer	AB,AC.	the record identified by the first ISN is to be read, values for fields AB and AC are to be returned
Search Buffer	AA.	
Value Buffer	C’87654321’ X’F8F7F6F5F4F3F2F1’	

Example 7

Find using coupled files. Select the records in file 1 containing the value "+100" for the field AB that are coupled to records in file 2 containing the value ’ABCDE’ for the field RB.

Control Block

Command Code	S1	
Command ID	EFGH	a non-blank command ID is used since ISN overflow may occur
File Number	1	file 1 is the primary file
ISN Lower Limit	0	select all qualifying ISNs
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	14	or larger
Value Buffer Length	12	or larger
ISN Buffer Length	100	return a maximum of 25 ISNs with each call
Command Option 1	b	save-ISN-list option is not to be used
Additions 3	password	file 2 is security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	/1/AB,D,/2/RB.	
Value Buffer	X’100CC1C2C3C4C54040404040’	

Because file 1 was specified as the primary file, the resulting ISNs will be from file 1. If ISNs from file 2 are also desired, the find may be repeated with file number 2 specified in the file number field. The order of the search criteria in the search buffer need not be changed.

Example 8

Find using multiple search criteria (complex search). Select the set of records in file 2 containing a value of "ABCD" for subdescriptor SA, a value less than "80" for field XB, and a value in the range "MMMMM" through "ZZZZZ" (but not "Sbbbb" through "TZZZZ") for field XE.

Control Block

Command Code	S1	
Command ID	GGGG	a non-blank command ID is used since the save-ISN-list option is to be used
File Number	2	
ISN Lower Limit	0	select all qualifying ISNs
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	35	or larger
Value Buffer Length	27	or larger
ISN Buffer Length	0	no ISNs to be returned in the ISN buffer
Command Option 1	H	save-ISN-list option is to be used
Additions 3	password	file 2 is security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	SA,D,XB,3,U,LT,D,XE,S,XE,N,XE,S,XE.	
Value Buffer	C’ABCD080MMMMMZZZZZSbbbbTZZZZ’	

S5 Command: Find Coupled ISNs

The S2 command returns or saves a list of coupled ISNs for the specified file.

This chapter covers the following topics:

- Function and Use
- Command: S5
- Control Block
- ISN Buffer
- Example

Function and Use

The S5 command is used to determine the records in one file that are coupled to a given record in another file.

The user specifies the file number and a given ISN within the file, plus the file number from which the coupled ISNs are to be returned. An optional ISN value above which ISNs are to be returned can also be specified.

Adabas determines which records are coupled to the specified record by using the Associator coupling lists. No access to Data Storage is required.

Adabas returns the resulting ISNs in the ISN buffer.

Command: S5

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	F	A
ISN LOWER LIMIT	17-20	binary	F	U
ISN QUANTITY	21-24	binary	--	A
	25-32	--	--	--
ISN BUFFER LENGTH	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	U
	45-48	--	--	--
ADDITIONS 3	49-56	alphanumeric	F	A
	57-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	*	--
RECORD BUFFER	*	--
SEARCH BUFFER	*	--
VALUE BUFFER	*	--
ISN BUFFER	--	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block

Command Code

S5

Command ID

A non-blank, non-zero value can be specified in this field if the Save ISN List option is to be used, or if overflow ISNs are to be stored on and then later read from the Adabas Work.

The first byte of this field may not be set to hexadecimal ’FF’.

File Number

The number of the file from which the coupled ISNs are to be selected. This file, called the "primary" file, must be coupled to the file specified in the additions 1 field, and cannot be an Adabas expanded file. The file number of physically coupled files must be no greater than 255.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Nonzero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

The ISN of the record for which the coupled ISNs are to be returned. The ISN must be present in the file specified in the additions 1 field. Adabas will return the first ISN of the resulting coupled ISN list in this field.

ISN Lower Limit

This field may be used in an initial Sx call to limit the resulting ISN list to those ISNs which are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from the Adabas Work.

ISN Quantity

An initial S5 call returns the number of records in the specified file that satisfy the search criteria.

In subsequent Sx calls used to retrieve ISNs from the Adabas Work, this field contains the number of ISNs placed in the ISN buffer.

ISN Buffer Length

The ISN buffer length (in bytes). This length is used to determine the number of ISNs placed in the ISN buffer.

If this field is set to zeros, no ISNs will be inserted in the ISN buffer.

To save the ISN list on Work for later processing, specify "H" in the command option 1 field and a valid command ID. The ISN buffer length field should be set to zeros if the resulting ISN list is to be read with the GET NEXT option of the L1/L4 command, or if the command is being issued only to determine the number of qualifying records.

If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas reduces the length to the next lower integer which is a multiple of 4.

Command Option 1: Save ISN List Option

Option	Description
H	(save-ISN-list) stores the entire ISN list resulting from an S5 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.

Command Option 1 or 2: Release Command ID Option

The "I" option may be specified in either the command option 1 or command option 2 field:

Option	Description
I	releases the command ID (CID) value specified in the command ID field. This is the first action taken during S5 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Additions 1: File Number

The number of the file which contains the ISN specified in the ISN field. The file number must be entered in the first two bytes of this field. The number must be provided in binary format. The remaining positions of this field must be set to blanks. This field must not be changed between successive S5 calls.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

ISN Buffer

Adabas places the list of resulting ISNs in the ISN buffer. Each ISN is returned as a four-byte binary number. The first ISN in the list is also returned in the control block's ISN field.

The ISNs are returned in ascending ISN sequence.

If the ISN buffer length is neither zero nor large enough to contain all the resulting ISNs, and a valid command ID was used, Adabas will store the overflow ISNs on the Adabas Work. These ISNs may then be retrieved using additional S5 calls that specify the same command ID. See the section *ISN List Processing* for additional information.

Example

Select the records in file 2 that are coupled to the record in file 1 identified with ISN 5. Use the save-ISN-list option.

Control Block

Command Code	S5	
Command ID	S501	a non-blank command ID is required if the save-ISN-list option is to be used
File Number	2	records to be selected from file 2
ISN	5	ISN 5 identifies the record for which the coupled records are to be selected
ISN Lower Limit	0	all qualifying ISNs are to be selected
ISN Buffer Length	0	no ISNs are to be returned in the ISN buffer
Command Option 1	H	save-ISN-list option to be used
Additions 1	X'0001404040404040'	the record for which the coupled records are to be selected is contained in file 1
Additions 3	password	file is security-protected

S8 Command: Process ISN Lists

The S8 command combines two ISN lists from the same file with an AND, OR, or NOT operation. For more information refer to the section *ISN List Processing*.

This chapter covers the following topics:

- Function and Use
- Command: S8
- Control Block
- ISN Buffer
- Example

Function and Use

The S8 command performs logical processing on two ISN lists that were previously created with Sx commands. Both ISN lists *must* be

- derived from the same file;
- in ISN sequence;
- stored on the Work dataset; and
- identified by command IDs assigned to the lists when they were created.

ISN lists resulting from an S2 or S9 command that are not in ascending ISN sequence cannot be used.

No activity (access or update) may be performed on the ISN lists to be processed between the time they are created and the time the S8 command is executed.

The S8 command may be used to perform the following logical operations:

Operator	**The resulting ISN list contains those ISNS that are present in . . .**
AND	both ISN lists
OR	either of the ISN lists
NOT	the first ISN list but not the second ISN list

The resulting ISNs are returned in the ISN buffer and/or stored on the Work dataset in ascending ISN sequence, depending on the specified command option and the command ID field setting:

- The resulting ISN list is saved in both the ISN buffer and on the Work dataset when a non-blank, non-zero command ID is specified and the save-ISN-list option is also specified.

- The resulting ISN list is saved in the ISN buffer *but not* on the Work dataset when no (or an invalid) command ID is specified with the save-ISN-list option.

Command: S8

User Control Block

Field	**Position**	**Format**	**Before Adabas Call**	**After Adabas Call**
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN LOWER LIMIT	17-20	binary	F	U
ISN QUANTITY	21-24	binary	--	A
	25-32	--	--	--
ISN BUFFER LENGTH	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	U
	45-48	--	--	--
ADDITIONS 3	49-56	alphanumeric	F	A
	57-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	*	--
RECORD BUFFER	*	--
SEARCH BUFFER	*	--
VALUE BUFFER	*	--
ISN BUFFER	--	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement

Control Block

Command Code

S8

Command ID

A nonblank, nonzero command ID must be specified in this field if a command option is specified in command option 1 field:

- The "I" (release command ID) option releases the specified command ID and any related ISN list as the first action taken during the S8 execution.
- With the "H" (save-ISN-list) option, the ISN list resulting from the S8 execution is stored under the specified command ID. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.

For more information refer to the section *ISN List Processing*.

The first byte of this field may not be set to hexadecimal ’FF’.

File Number

The number of the file from which both ISN lists to be processed were obtained.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

Adabas returns the first ISN of the resulting ISN list in this field. If there were no resulting ISNs, this field is not modified. This applies to both the initial call and any subsequent calls that are used to retrieve ISNs from the Adabas Work dataset.

ISN Lower Limit

This field may be used in an initial Sx call to limit the resulting ISN list to those ISNs that are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from the Adabas Work dataset.

ISN Quantity

As a result of an initial S8 call, this field returns the number of ISNs in the resulting ISN list.

As a result of a subsequent S8 call to retrieve ISNs from the Adabas Work dataset, this field contains the number of ISNs returned in the ISN buffer.

ISN Buffer Length

The ISN buffer length (in bytes). This length is used to determine the number of ISNs placed in the ISN buffer.

If this field is set to zeros, no ISNs are inserted in the ISN buffer. This field should be set to zeros if the resulting ISN list is to be read with the GET NEXT option of the L1/L4 command, or if the command is being issued only to determine the number of qualifying records.

If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas reduces the length to the next lower integer which is a multiple of 4.

Command Option 1: Save ISN List Option, Release Command ID Option

Option	Description
H	(save-ISN-list) stores the entire ISN list resulting from an S8 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.
I	releases the command ID (CID) value specified in the command ID field and any related ISN list as the first action taken during the S8 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Command Option 2: Logical Operator

The value entered in this field indicates the logical operation to be performed on the ISN lists:

Option	Operation	**The resulting ISN list contains those ISNs that are present in ...**
D	AND	both ISN lists.
O	OR	either ISN list.
N	NOT	the first ISN list but not the second ISN list.

Additions 1: Command IDs

The command IDs that identify the ISN lists to be processed must be specified in this field (four bytes per command ID). Each ISN list must be currently stored on the Adabas Work dataset and should contain ISNs from the same file.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

ISN Buffer

Adabas places the list of resulting ISNs in this buffer. Each ISN is returned as a four-byte binary number. The ISNs are returned in ISN sequence.

If the ISN buffer is too small to contain all the resulting ISNs and a non-blank, non-zero command ID was used, Adabas stores the overflow ISNs on the Work dataset. These ISNs can be retrieved with further Sx calls using the same command ID. For more information refer to the section *ISN List Processing*.

Example

Perform a logical OR operation between two ISN lists to produce a third ISN list that contains ISNs present in either list. The ISN lists to be processed were stored on the Adabas Work dataset under the command IDs "U020" and "U021". Store the resulting ISN list on the Adabas Work under the command ID "U999". Use the save-ISN-list option.

Control Block

Command Code	S8	
Command ID	U999	store the resulting ISN list under the command ID U999
ISN Lower Limit	0	select all of the resulting ISNs
ISN Buffer Length	0	no ISNs are to be returned in the ISN buffer
Command Option 1	H	use the save-ISN-list option
Command Option 2	O	perform an OR operation
Additions 1	U020U021	process the ISN lists identified by the command IDs U020 and U021
Additions 3	bbbbbbbb	file not security-protected

S9 Command: Sort ISN Lists

The S9 command sorts an ISN list in ascending ISN or descriptor-specified sequence.

This chapter covers the following topics:

- Function and Use
- Command: S9
- Control Block
- ISN Buffer
- Examples

Function and Use

The S9 command sorts an ISN list provided by the user or created by a previous Sx command. The ISN list to be sorted may either be in the ISN buffer or on the Adabas Work dataset (if the command ID assigned to the list when it was created is specified in the additions 4 field).

The ISN list may be sorted in the order of

- ISN values (ascending ISN sequence);
- from one to three user-specified descriptors.

The user may specify from one to three descriptors which are to be used to control the sort sequence. Either ascending or descending sequence may be specified.

The ISN list to be sorted must contain ISNs in ascending sequence, which implies that the list was not created by an S2 or an S9 command that specified the descriptor sequence option.

The resulting ISN list is either returned in the ISN buffer, or optionally stored on the Adabas Work dataset.

The S9 command can also be performed on Adabas expanded files.

Command: S9

User Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
COMMAND CODE	3-4	alphanumeric	F	U
COMMAND ID	5-8	alphanumeric	F	U
FILE NUMBER	9-10	binary	F	U
RESPONSE CODE	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN LOWER LIMIT	17-20	binary	F	U
ISN QUANTITY	21-24	binary	F	A
	25-32	--	--	--
ISN BUFFER LENGTH	33-34	binary	F	U
COMMAND OPTION 1	35	alphanumeric	F	U
COMMAND OPTION 2	36	alphanumeric	F	U
ADDITIONS 1	37-44	alphanumeric	F	U
	45-48	--	--	--
ADDITIONS 3	49-56	alphanumeric	F	A
ADDITIONS 4	57-64	alphanumeric	F	A
	65-72	--	--	--
COMMAND TIME	73-76	binary	--	A
USER AREA	77-80	--	--	U

User Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
FORMAT BUFFER	*	--
RECORD BUFFER	*	--
SEARCH BUFFER	*	--
VALUE BUFFER	*	--
ISN BUFFER	F	A

where:

F	Filled in by user before Adabas Call
A	Filled in by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block

Command Code

S9

Command ID

A non-blank, non-zero command ID may be specified in this field. This command ID applies only to the "I" (release command ID) or "H" (save-ISN-list) options; the command ID for obtaining an ISN list stored on Work must be specified in the additions 4 field.

The first byte of this field may not be set to a hexadecimal ’FF’.

File Number

The number of the file from which the ISN list to be sorted was obtained.

The S9 command can also be performed on Adabas expanded files.

Note:
When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code

Adabas returns the response code for the command in this field. Response code 0 indicates that the command executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the additions 2 field, are described in the *Adabas Messages and Codes* documentation.

ISN

Adabas returns the first ISN of the resulting ISN list in this field. If there are no resulting ISNs, this field is not modified. This applies to both the initial call and any subsequent calls that are used to retrieve ISNs from the Adabas Work dataset.

ISN Lower Limit

This field may be used in an initial Sx call to limit the resulting ISN list to those that are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

This field is also used when a group of overflow ISNs from a saved ISN list is being retrieved from the Adabas Work dataset.

ISN Quantity

If the ISN list to be sorted is being provided in the ISN buffer, this field must contain the number of ISNs that are to be sorted. If security-by-value is being used, a response code 1 is returned combined with the value 0 (one record found) or 1 (more than one record found) in this field. For more information, see the *Adabas Security Manual*.

For an initial S9 call, Adabas returns the number of records contained in the resulting ISN list. For any subsequent S9 call that retrieves ISNs from the Work dataset, Adabas returns the number of ISNs placed in the ISN buffer.

ISN Buffer Length

The ISN buffer length (in bytes) is used to determine the number of ISNs placed in the ISN buffer.

- If this field is set to zeros, no ISNs are inserted in the ISN buffer. Set this field to zeros and specify "H" in the command option 1 field if the resulting ISN list is to be read with the GET NEXT option of the L1/L4 command. If the S9 command is being issued only to determine the number of qualifying records, specify zero in this field and no command ID to prevent a sorted ISN list from being returned or stored.

- If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas reduces the length to the next lower integer that is a multiple of 4.

If the ISNs to be sorted are contained in the ISN buffer, this field must contain a value equal to or larger than the number of ISNs to be sorted, multiplied by 4. ISN overflow is stored on Work, and can be retrieved by a later Sx command with a command ID matching that specified in the command ID field.

Command Option 1: Save ISN List Option

Option	Description
H	(save-ISN-list) stores the entire ISN list resulting from an S9 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost. If the resulting ISN list is to be read with the GET NEXT option of the L1/L4 command, use this option with the ISN buffer length field set to to zeros. If this option is specified, command option 2 *cannot* specify the "I" (release command ID) option.

Command Option 2: Descending Option

Option	Description
D	(descending sequence) sorts the ISN list in descending sequence. This option may *not* be specified when sorting by ISN values.

If no command option 2 is specified for an S9 command, the ISN list is sorted in ascending sequence.

Command Option 1 or 2: Release Command ID Option

The "I" option may be specified in either the command option 1 or command option 2 field:

Option	Description
I	releases the command ID (CID) value specified in the command ID field as the first action taken during S9 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs. If the "H" (save-ISN-list) option is specified as command option 1, this option *cannot* be specified as command option 2.

Additions 1: Sort Sequence

The sort sequence to be used must be specified in this field.

The value "ISNbbbbb" indicates that the ISN values are to be used as the sorting sequence.

If the sort sequence is to be based on the values of one or more descriptors, the descriptors to be used must be specified in this field. One to three descriptors, subdescriptors, and/or superdescriptors may be specified. Phonetic descriptors and descriptors contained within a periodic group may not be specified. A multiple-value field may be specified, in which case the ISNs will be sorted according to the lowest value present within a given record. The descriptors are specified beginning with byte 1 (left justified). Any remaining positions must be set to blanks.

The number of ISNs that can be sorted depends on the size of the ADARUN parameters defined by the DBA. If this limit is exceeded, no sort is performed and response code 1 is returned.

Additions 3: Password

This field is used to provide an Adabas security or ADAESI password. If the database, file, or fields are security-protected, the user must provide a valid security or ADAESI password. Adabas sets the additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Command ID

If the ISN list to be sorted is contained on the Adabas Work dataset, the command ID under which the list is stored must be specified in the first 4 bytes of this field.

If the ISN list to be sorted is in the ISN buffer, this field must be set to blanks.

Adabas sets the additions 4 field to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

ISN Buffer

The ISN list to be sorted may be provided by the user in this buffer.

Adabas places the list of resulting ISNs in this buffer. Each ISN is returned as a four-byte binary number. If the ISN sort option is in effect ("ISN" is entered in the additions 1 field), the ISNs are returned in ascending ISN sequence. Otherwise, the ISNs are returned in the order of the values of the user-specified descriptor(s).

If the ISN buffer length is neither zero nor large enough to contain all the resulting ISNs and a valid command ID was specified, Adabas stores the overflow ISNs on the Adabas Work. These ISNs may then be retrieved using further S9 calls in which the same command ID is specified in the additions 4 field. For more information refer to the section *ISN List Processing*.

Examples

Example 1

Sort an ISN list contained in the ISN buffer in ISN sequence. Sort 622 ISNs.

Control Block

Command Code	S9	
Command ID	S901	a non-blank, non-zero command ID is required
File Number	1	derive the ISN list to be sorted from file 1
ISN Quantity	622	sort 622 ISNs
ISN Lower Limit	0	select all ISNs
ISN Buffer Length	2488	or larger; each ISN to be sorted requires 4 bytes
Command Option 1	H	use the save-ISN-list option
Command Option 2	b	use ascending sequence
Additions 1	ISNbbbbb	use the ISN values as the sorting sequence
Additions 3	bbbbbbbb	file not security-protected
Additions 4	bbbbbbbb	the ISN list to be sorted is contained in the ISN buffer

Buffer Areas

ISN Buffer	The ISNs to be sorted are provided in this buffer. Each ISN must be provided as a 4-byte binary number.

Example 2

Sort an ISN list stored on the Adabas Work. The command ID under which the ISN list is stored is "U066". Sort the list using the descriptors AA and AB as the major and minor sequence fields. Use the descending option.

Control Block

Command Code	S9	
Command ID	S902	a non-blank, non-zero command ID is required
File Number	1	derive the ISN list to be sorted from file 1
ISN Lower Limit	0	select all ISNs
ISN Buffer Length	0	no ISNs are to be returned in the ISN buffer
Command Option 1	H	use the save-ISN-list option
Command Option 2	D	use the descending sequence
Additions 1	AAABbbbb	use AA as the major sequence field and AB as the minor sequence field
Additions 3	bbbbbbbb	file not security-protected
Additions 4	U066bbbb	the ISN list to be sorted is stored on the Adabas Work under the command ID "U066"

Programming Examples

This part of the Adabas Command Reference documentation provides programming examples of direct Adabas calls in each host language.

The information is organized under the following headings:

- File Definitions Used in Examples
- Examples for Assembler
- Examples for COBOL
- Example for PL/I
- Example for FORTRAN

File Definitions Used in Examples

The following file definitions are used in the examples that follow in this documentation. Two file structures (files 1 and 2) are used.

The following tables show the structures of files 1 and 2 where

SF	is standard format
SL	is standard length

File 1

File 1 is neither security-protected nor ciphered. Certain examples in the documentation assume that file 1 has been coupled to file 2 using the descriptor AA as the basis for coupling.

Adabas Definition	Explanation
01,GA	Group GA, consisting of fields AA and AB.
02,AA,8,A,DE,NU	Elementary field AA; SL is 8 bytes, SF is alphanumeric, descriptor, null value suppression.
02,AB,2,P,DE,NU	Elementary field AB; SL is 2, SF is packed, descriptor, null value suppression.
01,AC,20,A,NU	Elementary field AC; SL is 20, SF is alphanumeric, null value suppression.
01,MF,3,A,MU,DE,NU	Multiple value field MF; SL is 3, SF is alphanumeric, descriptor, null value suppression.
01,GB,PE	Periodic group GB.
02,BA,1,B,DE,NU	Elementary field BA (within periodic group GB); SL is 1, SF is binary, descriptor, null value suppression.
02,BB,5,P,NU	Elementary field BB (within periodic group GB); SL is 5, SF is packed, null value suppression.
02,BC,10,A,NU	Elementary field BC (within periodic group GB); SL is 10, SF is alphanumeric, null value suppression.
01,GC,PE	Periodic group GC.
02,CA,7,A,DE,NU	Elementary field CA (within periodic group GC); SL is 7, SF is alphanumeric, descriptor, null value suppression.
02,CB,10,A,MU,NU SF	Multiple value field CB (within periodic group GC); SL is 10, is alphanumeric, null value suppression.

File 2

File 2 is security-protected. It is not ciphered. Certain examples in this documentation assume that file 2 is coupled to file 1 using field RA as the basis for coupling.

Adabas Definition	Explanation
01,RG	Group RG, consisting of all the fields in the record.
02,RA,8,A,DE,NU	Elementary field RA; SL is 8, SF is alphanumeric, descriptor, null value suppression.
02,RB,10,A,DE	Elementary field RB; SL is 10, SF is alphanumeric, descriptor.
02,GX	Group GX, consisting of the fields XA, XB, XC, XD, and XE.
03,XA,10,A	Elementary field XA; SL is 10, SF is alphanumeric.
03,XB,2,P,DE	Elementary field XB; SL is 2, SF is packed, descriptor.
03,XC,6,U	Elementary field XC; SL is 6, SF is unpacked.
03,XD,8,A,DE,NU	Elementary field XD; SL is 8, SF is alphanumeric, descriptor, null value suppression.
03,XE,5,A,DE,NU	Elementary field XE; SL is 5, SF is alphanumeric, descriptor, null value suppression.
SA=RA(1,4)	Subdescriptor SA; derived from bytes 1 through 4 of field RA, format is alphanumeric.
SB=RA(1,8),RB(1,4)	Superdescriptor SB; derived from bytes 1 through 8 of field RA and bytes 1 through 4 of field RB, format is alphanumeric.
SC=XB(1,2),XC(1,6)	Superdescriptor SC; derived from bytes 1 through 2 of field XB and bytes 1 through 6 of field XC, format is binary.

Examples for Assembler

This section contains examples of using direct Adabas calls in Assembler. The previously defined Adabas files defined are used in each example.

```
*** CONTROL BLOCK
      DS   0F
CB    DS   0CL80         USER CONTROL BLOCK
      DC   CL2' '        RESERVED FOR ADABAS USE
CCODE DC   CL2' '        COMMAND CODE
CID   DC   CL4' '        COMMAND ID
FNR   DC   H'0'          FILE NUMBER
RC    DC   H'0'          RESPONSE CODE
ISN   DC   F'0'          ISN
ISNLL DC   F'0'          ISN LOWER LIMIT
ISNQ  DC   F'0'          ISN QUANTITY
FBL   DC   H'100'        FORMAT BUFFER LENGTH
RBL   DC   H'250'        RECORD BUFFER LENGTH
SBL   DC   H'50'         SEARCH BUFFER LENGTH
VBL   DC   H'100'        VALUE BUFFER LENGTH
IBL   DC   H'20'         ISN BUFFER LENGTH
COPT1 DC   CL1' '        COMMAND OPTION 1
COPT2 DC   CL1' '        COMMAND OPTION 2
ADD1  DC   CL8' '        ADDITIONS 1
ADD2  DC   CL4' '        ADDITIONS 2
ADD3  DC   CL8' '        ADDITIONS 3
ADD4  DC   CL8' '        ADDITIONS 4
ADD5  DC   CL8' '        ADDITIONS 5
CTIME DC   F'0'          COMMAND TIME
UAREA DC   CL4' '        USER AREA
*
*
*** USER BUFFER AREAS
FB    DC   CL100' '      FORMAT BUFFER
RB    DC   CL250' '      RECORD BUFFER
SB    DC   CL50' '       SEARCH BUFFER
VB    DC   CL100' '      VALUE BUFFER
IB    DC   CL20' '       ISN BUFFER
* * *
```

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Issue Open Command

```
EXMP1 MVC  CCODE,=C'OP'            OP COMMAND
      MVC  RB(4),=C'ACC.'          ACCESS ONLY REQUESTED
      CALL ADABAS,(CB,FB,RB)       CALL ADABAS
      CLC  RC,=H'0'                CHECK RESPONSE CODE
      BNE  EX1ERR                  BRANCH IF NOT 0
```

Issue Find Command

```
      MVC  CCODE,=C'S1'            FIND COMMAND
      MVC  CID,=C'S101'            NONBLANK CID REQUIRED FOR
*                                  IDENTIFICATION OF THE LIST
      MVC  FNR,=H'2'               FILE 2
      MVC  ISNLL,=F'0'             ALL QUALIFYING ISNS DESIRED
      MVC  IBL,=H'0'               ISN BUFFER NOT REQUIRED
      MVI  FB,C'.'                 NO READ OF DATA STORAGE
      MVC  SB(7),=C'XB,3,U.'       SEARCH CRITERION
      MVC  VB(3),=C'099'           SEARCH VALUE
      CALL ADABAS,(CB,FB,RB,SB,VB) CALL ADABAS
      CLC  RC,=H'0'                CHECK RESPONSE CODE
      BNE  EX1ERR                  BRANCH IF NOT 0
      CLC  ISNQ,=F'0'              CHECK NUMBER OF ISNS FOUND
      BE   EX1EXIT                 BRANCH TO EXIT IF NO ISNS FOUND
```

Read Each Qualifying Record

```
EX1B  MVC  CCODE,=C'L1'            READ COMMAND
      MVC  ISN,=F'0'               BEGIN WITH 1ST ISN IN LIST
      MVI  COPT2,C'N'              GET NEXT OPTION TO BE USED
      MVC  FB(3),=C'RG.'           ALL FIELDS TO BE RETURNED
EX1C  CALL ADABAS,(CB,FB,RB)       CALL ADABAS
      CLC  RC,=H'0'                CHECK RESPONSE CODE
      BE   EX1D                    BRANCH IF RESPONSE CODE 0
      CLC  RC,=H'3'                CHECK IF ALL RECORDS READ
      BE   EX1EXIT                 BRANCH IF YES
      B    EX1ERR                  BRANCH TO ERROR ROUTINE
EX1D  . . .                        PROCESS RECORD . . .
      B    EX1C                    BRANCH TO READ NEXT RECORD
```

Error Routine

```
EX1ERR EQU *
*                                  DISPLAY ERROR MESSAGE
*      .                           TERMINATE USER PROGRAM
```

Issue Close Command

```
EX1EXIT MVC  CCODE,=C'CL'          CLOSE COMMAND
        CALL ADABAS,(CB)           CALL ADABAS
        CLC  RC,=H'0'              CHECK RESPONSE CODE
        BNE  EX1ERR                BRANCH IF NOT 0
```

Example 2

- All records in file 1 are to be read in physical sequential order.
- Each record read is to be updated with the following values:
  - Field AA = ABCDEFGH
  - Field AB = 500
- User is to have exclusive control of file 1.

Issue Open Command

```
EXMP2 MVC  CCODE,=C’OP’             OPEN COMMAND
      MVC  RB(6),=C’EXU=1.’         EXCLUSIVE CONTROL REQUESTED
      CALL ADABAS,(CB,FB,RB)        CALL ADABAS
      CLC  RC,=H’0’                 CHECK RESPONSE CODE
      BE   EX2A                     BRANCH IF RESPONSE CODE 0
      B    EX2ERR                   BRANCH IF NOT 0
```

Issue Read Physical Sequential Command

```
EX2A MVC  CID,=C’L201’              NONBLANK CID REQUIRED
     MVC  FNR,=H’1’                 FILE 1 TO BE READ
     MVC  ISN,=F’0’                 ALL RECORDS TO BE READ
     MVC  FB(3),=C’GA.’             VALUES FOR FIELDS AA AND AB
*                                  (GROUP GA) TO BE RETURNED
EX2B MVC  CCODE,=C’L2’              READ PHYS. SEQ.
     CALL ADABAS,(CB,FB,RB)         CALL ADABAS
     CLC  RC,=H’0’                  CHECK RESPONSE CODE
     BE   EX2C                      BRANCH IF RESPONSE CODE 0
     CLC  RC,=H’3’                  CHECK FOR END-OF-FILE
     BE   EX2EXIT                   BRANCH TO EXIT IF END-OF-FILE
     B    EX2ERR                    BRANCH TO ERROR ROUTINE
```

Update Record

- The same fields are to be updated as were read.
- The same CID and format buffer can be used for the update command.
- The ISN of the record to be updated is already in the ISN field as a result of the L2 command.

```
EX2C MVC  CCODE,=C’A1’              UPDATE COMMAND
     MVC  RB(8),=C’ABCDEFGH’        VALUE FOR FIELD AA
     MVC  RB+8(2),=PL2’500’         VALUE FOR FIELD AB
     CALL ADABAS,(CB,FB,RB)         CALL ADABAS
     CLC  RC,=H’0’                  CHECK RESPONSE CODE
     BE   EX2B                      BRANCH TO READ NEXT RECORD
```

Error Routine

```
EX2ERR EQU *
*      .                            DISPLAY ERROR MESSAGE
*      .                            TERMINATE USER PROGRAM
```

Close User Session

```
EX2EXIT MVC  CCODE,=C’CL’           CLOSE COMMAND
        CALL ADABAS,(CB)            CALL ADABAS
        CLC  RC,=H’0’               CHECK RESPONSE CODE
        BNE  EX2ERR                 BRANCH IF NOT 0
        . . .
```

Example 3 : User Session with ET Logic

During user session initialization, the user program is to display information indicating the last successfully processed transaction of the previous user session.

For each user transaction, the user program is to

- accept from a terminal 8 characters of input to be used as the key for updating files 1 and 2; and
- issue the Find command for file 1 to determine if a record exists with field AA = input key.

If no record is found, the user program is to issue a message. If a record is found, the user program is to

- delete the record from file 1; and
- add a new record to file 2: Field RA = input key entered.

Other fields are to contain a null value.

If the record cannot be successfully added, the user program is to issue a BT command and display an error message.

If both updates are successful, the user program is to issue an ET command.

Session Initialization

The section of the program illustrated is only executed during user session initialization:

Issue Open Command

The OP command is issued with ET data of the previous session being read:

```
EXMP3 EQU  *
      MVC  CCODE,=C'OP'            OPEN COMMAND
      MVI  COPT2,C'E'              ET DATA TO BE READ
      MVC  ADD1,=C'USER0001'       USER IDENTIFICATION
      MVC  ADD3,=C'PASSWORD'       USER PASSWORD
      MVC  RB(8),=C'UPD=1,2.'      FILES 1 AND 2 TO BE UPDATED
      CALL ADABAS,(CB,FB,RB)       CALL ADABAS
      CLC  RC,=H'0'                CHECK RESPONSE CODE
      BE   EX3A                    BRANCH IF RESPONSE CODE 0
      CLC  RC,=H'9'                CHECK FOR RESPONSE CODE 9
      BE   EXMP3                   BRANCH TO REPEAT OPEN
      B    EX3ERR                  BRANCH IF NOT 0 OR 9
EX3A  EQU  *
      CLC  CID,=F'0'               CHECK IF ET DATA FROM
*                                  PREVIOUS SESSION EXISTS
      BE   EX3B                    BRANCH IF NO ET DATA
*     . . .
```

Display ET Data

Display the ET data contained in the record buffer on the terminal screen to inform the user of the last successfully processed transaction of the previous user session:

```
      B    EX3C                    BRANCH TO BEGIN TRANS. PROCESS.
EX3B  EQU  *
```

No ET Data Received

If no ET data was received, a message is displayed indicating that no transactions were successfully processed during the previous user session.

Transaction Processing

This section is executed for each user transaction:

```
EX3C EQU  *
*    . . .                            ACCEPT INPUT FROM TERMINAL  . . .
```

Issue Find Command

Issue the Find command for file 1 to determine if a record exists with the field AA equal to the input key entered:

```
EX3D EQU  *
     MVC  CCODE,=C’S4’                FIND WITH HOLD COMMAND
     MVC  CID,=C’    ’                ISN LIST NOT TO BE SAVED
     MVC  FNR,=H’1’                   FILE 1
     MVC  ISNLL,=F’0’                 ALL QUALIFY. ISNS TO BE RETURNED
     MVI  FB,C’.’                     NO READ OF DATA STORAGE
     MVC  SB(3),=C’AA.’               SEARCH CRITERION
     MVC  VB(8),INPUT                 SEARCH VALUE
     CALL ADABAS,(CB,FB,RB,SB,VB,IB)  CALL ADABAS
     CLC  RC,=H’0’                    CHECK RESPONSE CODE
     BE   EX3E                        BRANCH IF RESPONSE CODE 0
     B    EX3ERR                      BRANCH TO ERROR ROUTINE
EX3E EQU  *
     CLC  ISNQ,=F’0’                  CHECK NUMBER OF RECORDS FOUND
     BNE  EX3F                        BRANCH IF RECORD FOUND
```

Issue Message if No Record is Found

If no record is found, the user program issues a message requesting a correction:

```
B    EX3C                        RETURN TO ACCEPT USER INPUT
```

Delete Record from File 1

The ISN of the record to be deleted is already in the ISN field and in hold status as a result of the S4 command.

```
EX3F EQU  *
     MVC  CCODE,=C’E4’                DELETE COMMAND
     CALL ADABAS,(CB)                 CALL ADABAS
     CLC  RC,=H’0’                    CHECK RESPONSE CODE
     BE   EX3G                        BRANCH IF RESPONSE CODE 0
     CLC  RC,=H’9’                    CHECK IF CURRENT TRANS. HAS BEEN
*                                     BACKED OUT BY ADABAS
     BE   EX3D                        IF YES, BRANCH TO REPEAT S4
     B    EX3ERR                      BRANCH TO ERROR ROUTINE
```

Add a New Record to File 2

```
EX3G EQU  *
     MVC  CCODE,=C’N1’                  ADD NEW RECORD
     MVC  FNR,=H’2’                     FILE 2
     MVC  FB(6),=C’RA.’                 VALUE BEING PROVIDED FOR RA
     MVC  RB(8),INPUT                   VALUE FOR FIELD RA
     CALL ADABAS,(CB,FB,RB)             CALL ADABAS
     CLC  RC,=H’0’                      CHECK RESPONSE CODE
     BE   EX3I                          BRANCH IF RESPONSE CODE 0
     CLC  RC,=H’9’                      WAS TRANSACTION BACKED OUT?
     BE   EX3D                          IF YES, RETURN TO REISSUE TRANS.
```

Unable to Add a New Record

If the attempt to add a new record is not successful, the transaction is backed out and the user is notified that an error condition exists.

```
     MVC  CCODE,=C’BT’                  BACKOUT TRANSACTION
     CALL ADABAS,(CB)                   CALL ADABAS
     CLC  RC,=H’0’                      CHECK IF RESPONSE CODE 0
     BE   EX3H                          BRANCH IF 0
```

Backout Not Successful

When the backout is not successful, a message is issued indicating that result.

```
     B    EX3ERR                        BRANCH TO ERROR ROUTINE
EX3H EQU  *
```

Backout Successful

When the backout is successful, a message is issued indicating that after an error was detected, the transaction was backed out.

```
     B    EX3ERR                        BRANCH TO ERROR ROUTINE
```

Updates Successfully Executed : Issue ET Command with ET Data

When the updates have been successfully executed, an ET command with ET data is issued.

```
EX3I EQU  *
     MVC  CCODE,=C’ET’        END OF TRANSACTION COMMAND
     MVI  COPT2,C’E’          ET DATA TO BE WRITTEN
     MVC  RB(8),INPUT         ET DATA CONSISTS OF INPUT KEY OF THIS TRANSACTION
     CALL ADABAS,(CB,FB,RB)   CALL ADABAS
     CLC  RC,=H’0’            CHECK RESPONSE CODE
     BE   EX3C                IF RESPONSE CODE 0, RETURN TO RECEIVE INPUT FOR
*                             THE NEXT TRANSACTION
     CLC  RC,=H’9’            CHECK IF CURRENT TRANSACTION HAS BEEN BACKED OUT
*                             BY ADABAS
     BE   EX3D                IF CURRENT TRANSACTION HAS BEEN BACKED OUT,
*                             RETURN TO REISSUE TRANSACTION
```

Error Routine

```
EX3ERR EQU  *
*      .                      NONZERO RESPONSE CODE RECEIVED
*      .                      DISPLAY ERROR MESSAGE
*      .                      TERMINATE USER PROGRAM
       . . .
INPUT  DS   CL8               KEY ENTERED FROM TERMINAL
```

Examples for COBOL

This section contains examples of using direct Adabas calls in COBOL. The previously defined Adabas files are used in each example.

```
*
 *** CONTROL BLOCK
     01  CONTROL-BLOCK.
         02 FILLER                     PIC X(2) VALUE SPACES.
         02 COMMAND-CODE               PIC X(2) VALUE SPACES.
         02 COMMAND-ID                 PIC X(4) VALUE SPACES.
         02 FILE-NUMBER                PIC S9(4) COMP VALUE +0.
         02 RESPONSE-CODE              PIC S9(4) COMP VALUE +0.
         02 ISN                        PIC S9(8) COMP VALUE +0.
         02 ISN-LOWER-LIMIT            PIC S9(8) COMP VALUE +0.
         02 ISN-QUANTITY               PIC S9(8) COMP VALUE +0.
         02 FORMAT-BUFFER-LENGTH       PIC S9(4) COMP VALUE +100.
         02 RECORD-BUFFER-LENGTH       PIC S9(4) COMP VALUE +250.
         02 SEARCH-BUFFER-LENGTH       PIC S9(4) COMP VALUE +50.
         02 VALUE-BUFFER-LENGTH        PIC S9(4) COMP VALUE +100.
         02 ISN-BUFFER-LENGTH          PIC S9(4) COMP VALUE +20.
         02 COMMAND-OPTION-1           PIC X VALUE SPACE.
         02 COMMAND-OPTION-2           PIC X VALUE SPACE.
         02 ADDITIONS-1                PIC X(8) VALUE SPACES.
         02 ADDITIONS-2                PIC X(4) VALUE SPACES.
         02 ADDITIONS-3                PIC X(8) VALUE SPACES.
         02 ADDITIONS-4                PIC X(8) VALUE SPACES.
         02 ADDITIONS-5                PIC X(8) VALUE SPACES.
         02 COMMAND-TIME               PIC S9(8) COMP VALUE +0.
         02 USER-AREA                  PIC X(4) VALUE SPACES.
*
*** USER BUFFER AREAS
    01 FORMAT-BUFFER                   PIC X(100) VALUE SPACES.
    01 RECORD-BUFFER                   PIC X(250) VALUE SPACES.
    01 SEARCH-BUFFER                   PIC X(50)  VALUE SPACES.
    01 VALUE-BUFFER                    PIC X(100) VALUE SPACES.
    01 ISN-BUFFER                      PIC X(20)  VALUE SPACES.
*

*** ADDITIONAL FIELDS USED IN THE EXAMPLES
    01 PROGRAM-WORK-AREA.
       05         COMM-ID PIC X(4).
       05         COMM-ID-X REDEFINES COMM-ID PIC S9(8) COMP.
       05         INPUT-KEY PIC X(8).
       05         RECORD-BUFFER-EX2.
          10      RECORD-BUFFER-A PIC X(8).
          10      RECORD-BUFFER-B PIC S9(3) COMP-3.
       05         RECORD-BUFFER-EX3.
          10      OPEN-RECORD-BUFFER.
             15   OPEN-RECORD-BUFFER-X PIC X(8).
             15   FILLER PIC S9(8) COMP.
          10      FILLER PIC X(18).
          10      UPDATED-XC PIC X(6).
          10      LAST-XD PIC X(8).
          10      FILLER PIC X(5).
       05         USER-DATA.
          10      RESTART-XD PIC X(8).
          10      RESTART-ISN PIC S9(8) COMP.
       05         SYNC-CHECK-SWITCH PIC 9 VALUE 0.
       05         AB-VALUE PIC S9(4) COMP-3 VALUE +500.
*
```

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Issue Open Command

```
EXMP1.
        MOVE      'OP' TO COMMAND-CODE.
        MOVE      'ACC.' TO RECORD-BUFFER.
        CALL      'ADABAS'
                  USING CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX1ERR.
```

Issue Find Command

```
MOVE      'S1' TO COMMAND-CODE.
        MOVE      'S101' TO COMMAND-ID.
        MOVE      2 TO FILE-NUMBER.
        MOVE      0 TO ISN-LOWER-LIMIT.
        MOVE      0 TO ISN-BUFFER-LENGTH.
        MOVE      '.' TO FORMAT-BUFFER.
        MOVE      'XB,3,U.' TO SEARCH-BUFFER.
        MOVE      '099' TO VALUE-BUFFER.
        CALL      'ADABAS' USING CONTROL-BLOCK, FORMAT-BUFFER,
                  RECORD-BUFFER, SEARCH-BUFFER, VALUE-BUFFER.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX1ERR.
    EX1A.
        IF ISN-QUANTITY = 0  GO TO EX1EXIT.
```

Read Each Qualifying Record

```
EX1B.
        MOVE      'L1' TO COMMAND-CODE.
        MOVE      0 TO ISN.
        MOVE      'N' TO COMMAND-OPTION-2.
        MOVE      'RG.' TO FORMAT-BUFFER.
    EX1C.
        CALL      'ADABAS'
                  USING CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE = 0 GO TO EX1D.
        IF RESPONSE-CODE = 3 GO TO EX1EXIT.
    EX1D.
        . . . PROCESS RECORD . . .
        GO TO EX1C.
```

Error Routine

```
EX1ERR.
*       .DISPLAY ERROR MESSAGE
*       .TERMINATE USER PROGRAM
```

Issue Close Command

```
EX1EXIT.
        MOVE      ’CL’ TO COMMAND-CODE.
        CALL      ’ADABAS’ USING CONTROL-BLOCK.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX1ERR.
```

Example 2

- All records in file 1 are to be read in physical sequential order.
- Each record read is to be updated with the following values:
  - Field AA = ABCDEFGH
  - Field AB = 500
- User is to have exclusive control of file 1.

Issue Open Command

```
EXMP2.
        MOVE      ’OP’ TO COMMAND-CODE.
        MOVE      ’EXU=1.’ TO RECORD-BUFFER.
        CALL      ’ADABAS’ USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX2ERR.
```

Issue Read Physical Sequential Command

```
EX2A.
        MOVE      ’L201’ TO COMMAND-ID.
        MOVE      1 TO FILE-NUMBER.
        MOVE      0 TO ISN.
        MOVE      ’GA.’ TO FORMAT-BUFFER.
    EX2B.
        MOVE      ’L2’ TO COMMAND-CODE.
        CALL      ’ADABAS’ USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE = 0  GO TO EX2C.
        IF RESPONSE-CODE = 3  GO TO EX2EXIT.
        GO TO EX2ERR.
```

Update Record

- The same fields are to be updated as were read.
- The same CID and format buffer can be used for the update command.
- The ISN of the record to be updated is already in the ISN field as a result of the L2 command.

```
EX2C.
        MOVE      'A1' TO COMMAND-CODE.
        MOVE      'ABCDEFGH' TO RECORD-BUFFER-A.
        MOVE      AB-VALUE TO RECORD-BUFFER-B.
        CALL      'ADABAS' USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER-EX2.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX2ERR.
        GO TO EX2B.
```

Error Routine

```
EX2ERR.
        . DISPLAY ERROR MESSAGE
        . TERMINATE USER PROGRAM
```

Close User Session

```
EX2EXIT.
        MOVE      'CL' TO COMMAND-CODE.
        CALL      'ADABAS' USING CONTROL-BLOCK.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX2ERR.
```

Example 3 : User Session with ET Logic

During user session initialization, the user program is to display information indicating the last successfully processed transaction of the previous user session.

For each user transaction, the user program is to

- accept from a terminal 8 characters of input to be used as the key for updating files 1 and 2; and
- issue the Find command for file 1 to determine if a record exists with field AA = input key.

If no record is found, the user program is to issue a message. If a record is found, the user program is to

- delete the record from file 1; and
- add a new record to file 2: Field RA = input key entered.

Other fields are to contain a null value.

If the record cannot be successfully added, the user program is to issue a BT command and display an error message.

If both updates are successful, the user program is to issue an ET command.

Session Initialization

This section of the program is only executed during user session initialization.

- The OP command is issued with ET data of the previous session being read.
- A message is displayed on the terminal screen identifying the last successfully processed transaction of the user's previous session.

```
    EX3.
         MOVE    ’OP’ TO COMMAND-CODE.
         MOVE    ’E’ TO COMMAND-OPTION-2.
         MOVE    ’USER0002’ TO ADDITIONS-1.
         MOVE    ’PASSWORD’ TO ADDITIONS-3.
         MOVE    ’UPD=1,2.’ TO RECORD-BUFFER.
         CALL    ’ADABAS’ USING
                 CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
         IF RESPONSE-CODE = 9  GO TO EX3.
         IF RESPONSE-CODE NOT EQUAL TO 0
              GO TO EX3ERR.
    EX3A.
         MOVE    COMMAND-ID TO COMM-ID.
         IF COMM-ID-X = +0
              GO TO EX3B.
* Display ET data (contained in RECORD BUFFER) on screen to inform user of
* last successfully processed transaction of previous user session.
                . . .DISPLAY ET DATA. . .
              GO TO EX3C.
    EX3B.
*** No ET data received.
*   Display message that no transactions were successfully processed during
*   the previous user session
                . . .DISPLAY MESSAGE . . .
*** Transaction processing.
*   This section is executed for each user transaction.
    EX3C.
*               . . .ACCEPT INPUT FROM TERMINAL. . .
*   Issue Find command for file 1 to determine if record exists with field AA
*   equal to input key entered.
    EX3D.
         MOVE    ’S4’ TO COMMAND-CODE.
         MOVE    SPACES TO COMMAND-ID.
         MOVE    1 TO FILE-NUMBER.
         MOVE    0 TO ISN-LOWER-LIMIT.
         MOVE    ’.’ TO FORMAT-BUFFER.
         MOVE    ’AA.’ TO SEARCH-BUFFER.
         MOVE    INPUT-KEY TO VALUE-BUFFER.
         CALL    ’ADABAS’ USING
                 CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER,
                 SEARCH-BUFFER, VALUE-BUFFER, ISN-BUFFER.
         IF RESPONSE-CODE = 0
              GO TO EX3E.
         GO TO EX3ERR.
EX3E.
         IF ISN-QUANTITY NOT EQUAL TO ZEROS
              GO TO EX3F.
***No records found, issue message requesting correction.
              . . .ISSUE MESSAGE . . .
         GO TO EX3C.
*** Delete record from file 1.
*  ISN of record to be deleted is already in ISN field and in hold
status
*  as a result of the S4 command.
    EX3F.
         MOVE     E3’ TO COMMAND-CODE.
         CALL     ’ADABAS’ USING CONTROL-BLOCK.
         IF RESPONSE-CODE = 0
              GO TO EX3G.
         IF RESPONSE-CODE = 9
              GO TO EX3D.
         GO TO EX3ERR.
```

```
*** Add new record to file 2.
    EX3G.
         MOVE     ’N1’ TO COMMAND-CODE.
         MOVE     2 TO FILE-NUMBER.
         MOVE     ’RA.’ TO FORMAT-BUFFER.
         MOVE     INPUT-KEY TO RECORD-BUFFER.
         CALL     ’ADABAS’ USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
         IF RESPONSE-CODE = 0
              GO TO EX3I.
         IF RESPONSE-CODE = 9
              GO TO EX3D.
*** Attempt to add new record not successful.
*  Backout transaction.
*  Notify user that error condition exists.
         MOVE     ’BT’ TO COMMAND-CODE.
         CALL     ’ADABAS’ USING control-block.
         IF RESPONSE-CODE = 0
              GO TO EX3H.
*** Backout not successful.
*   Issue message indicating that the backout was not successful
         GO TO EX3ERR.
    EX3H.
*** Backout successful.
*   Issue message indicating the error condition detected while while
adding a
*   new record
         GO TO EX3ERR.
*** Updates successfully executed.
*  Issue ET command with ET data.
  EX3I.
         MOVE     ’ET’ TO COMMAND-CODE.
         MOVE     ’E’ TO COMMAND-OPTION-2.
         MOVE     INPUT-KEY TO RECORD-BUFFER.
         CALL     ’ADABAS’ USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
         IF RESPONSE-CODE = 0
              GO TO EX3C.
         IF RESPONSE-CODE = 9
              GO TO EX3D.
*** Error Routine
    EX3ERR.
*        . DISPLAY ERROR MESSAGE
*        . TERMINATE USER PROGRAM
                . . .
```

Example for PL/I

This section contains examples of using direct Adabas calls in PL/ I . The previously defined Adabas files are used in each example.

```
/*** CONTROL BLOCK ***/
DCL 1    CONTROL_BLOCK,
    02   FILLER1               CHAR (2) INIT (’ ’),
    02   COMMAND_CODE          CHAR (2) INIT (’ ’),
    02   COMMAND_ID            CHAR (4) INIT (’ ’),
    02   FILE_NUMBER           BIN FIXED (15) INIT (0),
    02   RESPONSE_CODE         BIN FIXED (15) INIT (0),
    02   ISN                   BIN FIXED (31) INIT (0),
    02   ISN_LOWER_LIMIT       BIN FIXED (31) INIT (0),
    02   ISN_QUANTITY          BIN FIXED (31) INIT (0),
    02   FORMAT_BUFFER_LENGTH  BIN FIXED (15) INIT (100),
    02   RECORD_BUFFER_LENGTH  BIN FIXED (15) INIT (250),
    02   SEARCH_BUFFER_LENGTH  BIN FIXED (15) INIT (50),
    02   VALUE_BUFFER_LENGTH   BIN FIXED (15) INIT (100),
    02   ISN_BUFFER_LENGTH     BIN FIXED (15) INIT (20),
    02   COMMAND_OPTION_1      CHAR(1) INIT (’ ’),
    02   COMMAND_OPTION_2      CHAR(1) INIT (’ ’),
    02   ADDITIONS_1           CHAR(8) INIT (’ ’),
    02   ADDITIONS_2           CHAR(4) INIT (’ ’),
    02   ADDITIONS_3           CHAR(8) INIT (’ ’),
    02   ADDITIONS_4           CHAR(8) INIT (’ ’),
    02   ADDITIONS_5           CHAR(8) INIT (’ ’),
    02   COMMAND_TIME          BIN FIXED (31) INIT (0),
    02   USER_AREA             CHAR(4) INIT (’ ’);

/*** USER BUFFER AREAS ***/
DCL FORMAT_BUFFER    CHAR(100),
    RECORD_BUFFER    CHAR(250),
    SEARCH_BUFFER    CHAR(50),
    VALUE_BUFFER     CHAR(100),
    ISN_BUFFER       CHAR(20);
*
*

/***    ADDITIONAL FIELDS USED IN THE EXAMPLES ***/
DCL
    COMM_ID_X  BIN FIXED(31);
    COMM_ID    CHAR(4) BASED (ADDR(COMM_ID_X));
DCL      INPUT_KEY CHAR(8);
DCL      SYNC_CHECK_SWITCH CHAR(1) INIT(’0’);
DCL 1 RECORD_BUFFER_EX2,
         2   RECORD_BUFFER_A  CHAR(8),
         2   RECORD_BUFFER_B  DEC FIXED(3,0),
         2   FILLER3 CHAR(240);
DCL 1 RECORD_BUFFER_EX3,
         2   OPEN_RECORD_BUFFER,
             3   OPEN_RECORD_BUFFER_X CHAR(8),
             3   FILLER4 BIN FIXED(31),
         2   FILLER5 CHAR(18),
         2   UPDATED_XC CHAR(6),
         2   LAST_XD CHAR(8),
         2   FILLER6 CHAR(5),
    1 USER_DATA,
         2   RESTART_XD CHAR(8),
         2   RESTART_ISN BIN FIXED(31);
DCL      ADABAS ENTRY OPTIONS(ASM);
```

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Issue Open Command

```
*** Issue Open Command **/
EXMP1:
    COMMAND_CODE = ’OP’;
    RECORD_BUFFER = ’ACC.’;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE > 0        THEN GOTO EX1ERR;
```

Issue Find Command

```
/*** Issue Find Command ***/
    COMMAND_CODE = ’S1’;
    COMMAND_ID = ’S101’;
    FILE_NUMBER = 2;
    ISN_LOWER_LIMIT = 0;
    ISN_BUFFER_LENGTH = 0;
    FORMAT_BUFFER = ’.’;
    SEARCH_BUFFER = ’XB,3,U.’;
    VALUE_BUFFER = ’099’;
    CALL ADABAS (CONTROL_BLOCK, FORMAT_BUFFER,
         RECORD_BUFFER, SEARCH_BUFFER, VALUE_BUFFER);
    IF RESPONSE_CODE > 0 THEN GOTO EX1ERR;
EX1A:
    IF ISN_QUANTITY = 0 THEN GOTO EX1EXIT;
EX1B:
    COMMAND_CODE = ’L1’;
    ISN = 0;
    COMMAND_OPTION_1 = ’N’;
    FORMAT_BUFFER  = ’RG.’;
EX1C:
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX1D;
    IF RESPONSE_CODE = 3 THEN
         GOTO EX1EXIT;
EX1D:
    . . .PROCESS RECORD . . .
         GOTO EX1C;
```

Error Routine

```
/*** Error Routine ***/
EX1ERR:
/*   . DISPLAY ERROR MESSAGE */
/*   . TERMINATE USER PROGRAM */
```

Issue Close Command

```
/** Issue Close Command **/
EX1EXIT:
    COMMAND_CODE = ’CL’;
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE > 0 THEN
         GOTO EX1ERR;
```

Example 2

- All records in file 1 are to be read in physical sequential order.
- Each record read is to be updated with the following values:
    - Field AA = ABCDEFGH
    - Field AB = 500
- User is to have exclusive control of file 1.

Issue Open Command

```
/*** Issue Open Command ***/
EXMP2:
    COMMAND_CODE = ’OP’;
    RECORD_BUFFER = ’EXU=1.’;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE > 0 THEN   GOTO EX2ERR;
```

Issue Read Physical Sequence Command

```
/*** Issue Read Physical Seq.  Command ***/
EX2A:
    COMMAND_ID = ’L201’;
    FILE_NUMBER = 1;
    ISN = 0;
    FORMAT_BUFFER = ’GA.’;
EX2B:
    COMMAND_CODE = ’L2’;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN   GOTO EX2C;
    IF RESPONSE_CODE = 3 THEN   GOTO EX2EXIT;
    GOTO EX2ERR;
```

Update Record

```
/*** Update record.  ***/
/* Same fields are to be updated as were read.  */
/* Same CID and FORMAT BUFFER can be used for update.  */
/* ISN of record to be updated is already in ISN field as a result of */
/* the L2 command.  */
EX2C:
    COMMAND_CODE = ’A1’;
    RECORD_BUFFER_A = ’ABCDEFGH’;
    RECORD_BUFFER_B = 500;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,
                 RECORD_BUFFER_EX2);
    IF RESPONSE_CODE > 0 THEN   GOTO EX2ERR;
    GOTO EX2B;
```

Error Routine

```
/*** Error Routine ***/
EX2ERR:
/*   . DISPLAY ERROR MESSAGE */
/*   . TERMINATE USER PROGRAM */
```

Close User Session

```
/* Close User Session */
EX2EXIT:
    COMMAND_CODE = ’CL’;
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE > 0 THEN   GOTO EX2ERR;
```

Example 3

This example illustrates a user session with ET logic. The user program is to perform the following functions:

1. During user session initialization, display information indicating the last successfully processed transaction of the previous user session.

2. For each user transaction:

- Accept from a terminal 8 characters of input that is used as the key for updating files 1 and 2.
- Issue a Find command for file 1 to determine if a record exists with field AA = input key.
- If no record is found, issue a message.
- If a record is found:
  - Delete the record from file 1;
  - Add a new record to file 2: Field RA = input key entered. Other fields to contain null value.
  - If the record cannot be successfully added, issue a BT command, display error message.
  - If both updates are successful, issue an ET command.

Session Initialization

This section of the program is only executed during user session initialization.

- The OP command is issued with ET data of the previous session being read.
- A message is displayed on the terminal screen identifying the last successfully processed transaction of the user’s previous session.

```
EX3:
    COMMAND_CODE = 'OP';
    COMMAND_OPTION_2 = 'E';
    ADDITIONS_1 = 'USER0003';
    ADDITIONS_3 = 'PASSWORD';
    RECORD_BUFFER = 'UPD=1,2.';
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 9 THEN   GOTO EX3;
    IF RESPONSE_CODE > 0 THEN
         GOTO EX3ERR;
EX3A:
    COMM_ID = COMMAND_ID;
    IF COMM_ID_X = 0 THEN
         GOTO EX3B;
/*  Display ET data (contained in RECORD BUFFER) on screen to inform user of
    last successfully processed transaction of previous user session. */
         . . .DISPLAY ET DATA. . .
    GOTO EX3C;
EX3B:
/*                          */
/*** No ET data received.  */
/*   Display message that no transactions were successfully processed during
    the previous user session. */
         . . .DISPLAY MESSAGE . . .
/*                               */
/*** Transaction processing.  ***/
/* This section is executed for each user transaction.  */
EX3C:
         . . .ACCEPT INPUT FROM TERMINAL. . .
/*                                                       */
/* Issue Find command for file 1 to determine if rec exists with field AA
   equal to input key entered. */
EX3D:
    COMMAND_CODE = 'S4';
    COMMAND_ID = ' ';
    FILE_NUMBER = 1;
    ISN_LOWER_LIMIT = 0;
    FORMAT_BUFFER = '.';
    SEARCH_BUFFER = 'AA.';
    VALUE_BUFFER = INPUT_KEY;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER,
         SEARCH_BUFFER,VALUE_BUFFER,ISN_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3E;
    GOTO EX3ERR;
EX3E:
    IF ISN_QUANTITY > 0 THEN
         GOTO EX3F;
/*                                                       */
/* No record found, issue message requesting correction.  */
         . . .ISSUE MESSAGE . . .
    GOTO EX3C;
/*                              */
/* Delete record from file 1.  */
/* ISN of record to be deleted is already in ISN field and in hold
status
   as a result of the S4 command. */
EX3F:
    COMMAND_CODE = 'E4';
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3G;
    IF RESPONSE_CODE = 9 THEN
         GOTO EX3D;
    GOTO EX3ERR;
/***Add new record to file 2.  */
EX3G:
    COMMAND_CODE = ’N1’;
    FILE_NUMBER = 2;
    FORMAT_BUFFER = ’RA.’;
    RECORD_BUFFER = INPUT_KEY;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3I;
    IF RESPONSE_CODE = 9 THEN
         GOTO EX3D;
/*                                */
/*  Attempt to add new record not successful. Backout transaction and
notify
    user that error condition exists. */
    COMMAND_CODE = ’BT’;
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3H;
/*                            */
/*  Backout not successful.   */
     . . .ISSUE MESSAGE INDICATING BACKOUT NOT SUCCESSFUL . .
    GO TO EX3ERR.
/*                               */
EX3H:
/*** Backout successful.    ***/
/* Issue message indicating error condition detected while adding new
record.*/
         . . .ISSUE MESSAGE. . .
    GOTO EX3ERR;
/*                                    */
/*** Updates successfully executed.  ***/
/*   Issue ET command with ET data.    */
EX3I:
    COMMAND_CODE = ’ET’;
    COMMAND_OPTION_2 = ’E’;
    RECORD_BUFFER = INPUT_KEY;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3C;
    IF RESPONSE_CODE = 9 THEN
         GOTO EX3D;
/*                                   */
/***    Error Routine    ***/
EX3ERR:
/*  . DISPLAY ERROR MESSAGE */
/*  . TERMINATE USER PROGRAM */
         . . .
```

Example for FORTRAN

This section contains an example of using direct Adabas calls in FORTRAN. The previously defined Adabas files are used in each example.

```
C   *** CONTROL BLOCK ***
    INTEGER*4     CB(20),CID,ISN,ISNL,ISNQ
    INTEGER*4     ADD1(2),ADD2,ADD3(2),ADD4(2),ADD5(2)
    INTEGER*4     CTIME,UAREA
    INTEGER*2     CBI(40),CCODE,FNR,RC,FBL,RBL,SBL,VBL,IBL
    LOGICAL*1     CBL(80),COPT1,COPT2
    EQUIVALENCE       (CB(1),CBI(1),CBL(1))
    EQUIVALENCE       (CID,CB(2)),(ISN,CB(4))
    EQUIVALENCE       (ISNL,CB(5)),(ISNQ,CB(6))
    EQUIVALENCE       (ADD1(1),CB(10)),(ADD2,CB(12)),(ADD3(1),CB(13))
    EQUIVALENCE       (ADD4(1),CB(15),(ADD5(1),CB(17))
    EQUIVALENCE       (CTIME,CB(19)),(UAREA,CB(20))
    EQUIVALENCE       (CCODE,CBI(2)),(FNR,CBI(5)),(RC,CBI(6))
    EQUIVALENCE       (FBL,CBI(13)),(RBL,CBI(14)),(SBL,CBI(15))
    EQUIVALENCE       (VBL,CBI(16)),(IBL,CBI(17))
    EQUIVALENCE       (COPT1,CBL(35)),(COPT2,CBL(36))


C   *** USER BUFFER AREAS ***
    INTEGER*4 FB(25),RB(50),SB(10),VB(10),IB(50)
 *
 *
C   *** ADDITIONAL FIELDS USED IN THIS EXAMPLE  ***
    LOGICAL*1 BLANK/1H /,COPH/1HH/,PERIOD/1H./,COPN/1HN/
    INTEGER*2 S1/2HS1/,L1/2HL1/,CL/2HCL/
    INTEGER*4 CID1/4HS101/,FB1/4H.  /,FB2/4HRG. /,SB1/4HXB,3/
INTEGER*4 SB2/4H,U. /,VB1/4H099 /
```

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Initialize Control Block

```
c*** Initialize Control Block
    DO 5 I=1,80
    CBL(I)=BLANK
5   CONTINUE
    DO 10 I=3,6
    CB(I)=0
10  CONTINUE
    CBI(13)=100
    CBI(14)=200
    CBI(15)=40
    CBI(16)=40
    CBI(17)=200
    CBI(19)=0
```

Issue Find Command

```
c***Issue FIND Command
    CCODE=S1
    CID=CID1
    FNR=2
    ISNL=0
    COPT1=COPH
    FB(1)=FB1
    SB(1)=SB1
    SB(2)=SB2
    VB(1)=VB1
    CALL ADABAS(CB,FB,RB,SB,VB,IB)
    IF(RC.NE.0) GO TO 50
    IF(ISNQ.EQ.0) GO TO 100
```

Read Each Record Selected

```
c***Read Each Record Selected
15  CONTINUE
    CCODE=L1
    ISN=0
    COPT1=COPN
    FB(1)=FB2
    CALL ADABAS(CB,FB,RB)
    IF(RC.EQ.0) GO TO 30
    IF(RC.EQ.3) GO TO 100
    PRINT 60,RC,CCODE
60  FORMAT(1H0,'ADABAS ERROR CODE',I4,' FROM '.A2,' COMMAND')
    GO TO 50
30  CONTINUE
C   ...PROCESS RECORD...
    GO TO 15
50  CONTINUE
    STOP
100 CONTINUE
    ...
```